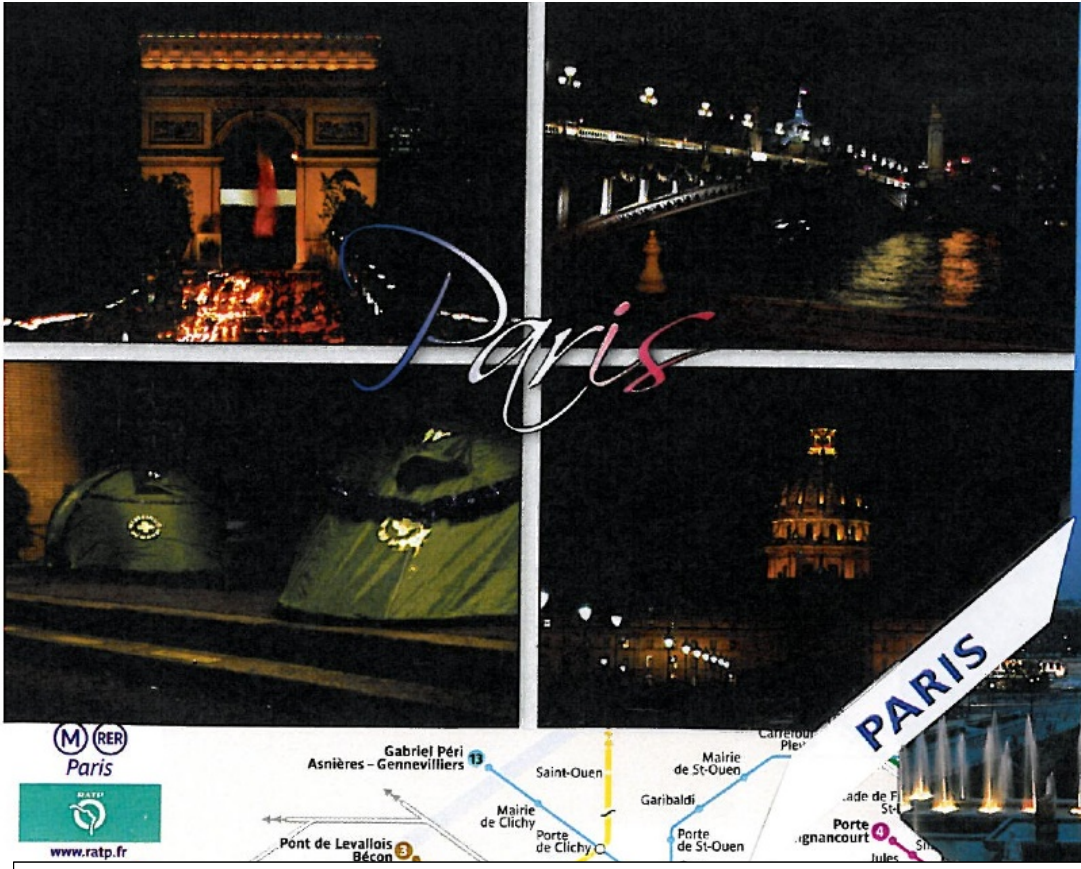
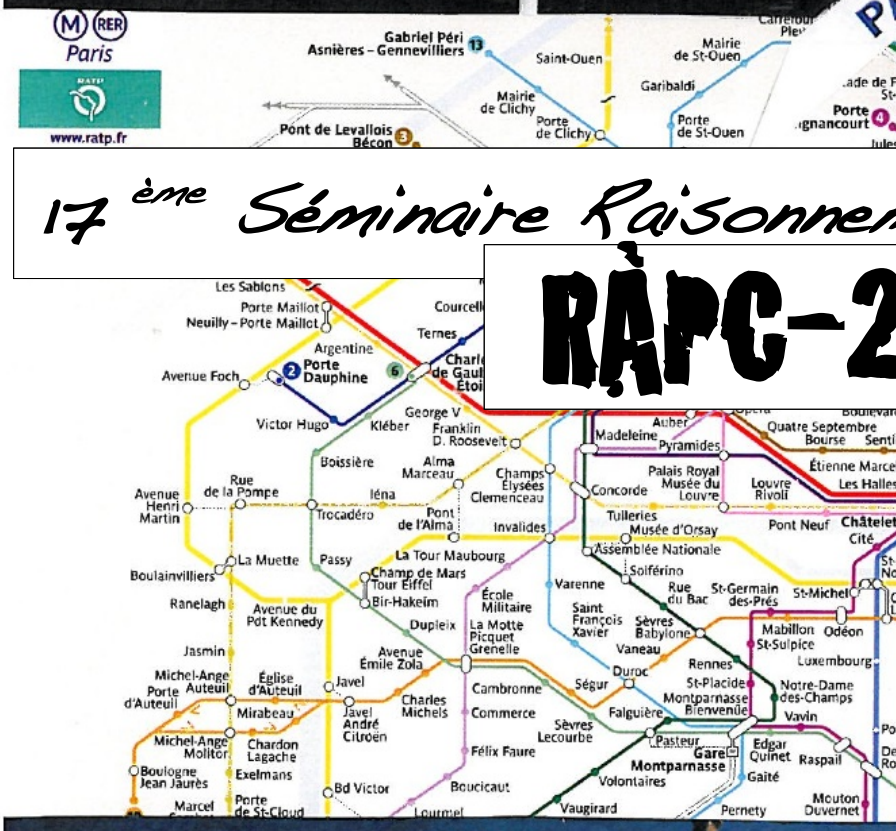
Actes du 17ème séminaire sur le raisonnement à partir de Cas, Paris (29–30 juin)

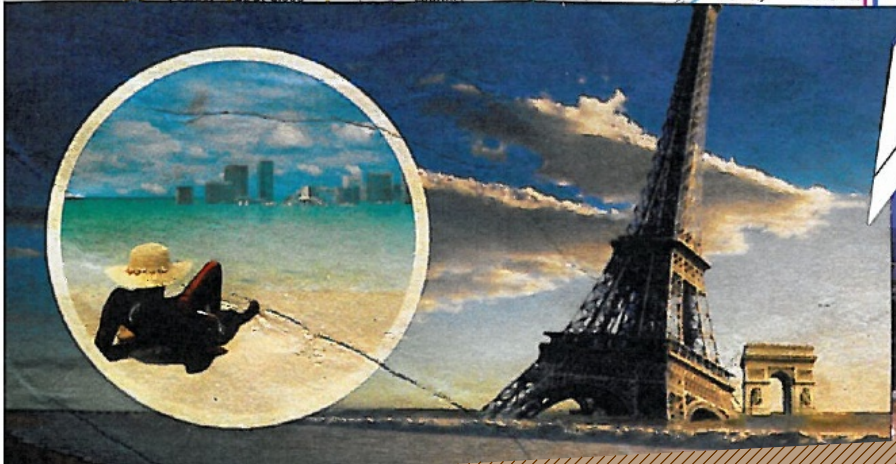
Béatrice Fuchs, Amedeo Napoli





17ème Séminaire Raisonnement à Partir de Cas

RÀPC-2009



Paris, 29-30 juin 2009

Béatrice Fuchs, Amedeo Napoli, éditeurs

PRÉAMBULE

L'atelier Raisonnement à Partir de Cas (RàPC) est un peu le rendez-vous obligé des chercheurs francophones sur le raisonnement à partir de cas. C'est un moment d'échange et de rencontre entre des chercheurs débutants ou confirmés, qui se veut ouvert vers l'extérieur, en particulier aux autres communautés scientifiques intéressées comme aux industriels désireux de partager leurs expériences. Cette année, le $17^{ième}$ atelier est organisé sous la direction scientifique de Béatrice Fuchs (LIRIS Lyon) et Amedeo Napoli (LORIA Nancy), et accueilli dans les locaux du SPIM à Paris grâce et avec l'aide bienveillante de Marie-Christine Jaulent (SPIM Paris) et Rim Bentebibel (LIPN, Université de Paris 13).

Le comité de programme a retenu treize propositions de communication réparties en six grandes sessions détaillées ci-après. Deux sessions sont consacrées à des travaux autour du projet fédérateur Taaable, une session va aux applications et une session porte sur les connaissances et les ontologies. En outre, cette année est un peu spéciale car elle verra également une rencontre longtemps attendue entre spécialistes du RàPC et spécialistes de l'analogie. Ainsi, tout un après midi sera dédié à l'étude des rapports entre ces deux disciplines très proches l'une de l'autre dans l'esprit mais pas nécessairement dans les méthodes.

- La première session « Au tour de Taaable (1) » regroupe quatre propositions relatives au projet Taaable, un système de RàPC textuel dans le domaine des recettes de cuisine en compétition au *Computer Cooking Contest*. Différents aspects de WikiTaaable, une extension de Taaable y sont étudiés : acquisition et extraction de connaissances (F. Badra, J. Cojan, A. Cordier, J. Lieber, T. Meilender, A. Mille, P. Molli, E. Nauer, A. Napoli, H. Skaf-Molli, Y. Toussaint), découverte opportuniste de connaissances d'adaptation (F. Badra, A. Cordier, J. Lieber), gestion des connaissances utilisant un wiki sémantique comme un tableau noir (A. Cordier, J. Lieber, P. Molli, E. Nauer, H. Skaf-Molli, Y. Toussaint) et mécanismes du moteur de raisonnement (J. Lieber).
- Une session spéciale sur l'analogie donne l'occasion d'échanges sur les liens « ancestraux » entre analogie et RàPC, avec une proposition de A. Ben Hassena et L. Miclet qui étudient la dissemblance analogique comme une mesure de dissimilarité entre structures arborescentes. D'autres exposés sont prévus — Y. Lepage (GREYC Caen), H. Prade (IRIT Toulouse) et L. Miclet (IRISA ENSSAT Lannion) — dans cette session qui va rassembler les communautés RàPC et analogie et qui se terminera par une discussion globale.
- Une session « travaux préliminaires » donnera la possibilité à des chercheurs débutants de s'exprimer et d'échanger des vues sur leurs travaux de recherche.
- Une quatrième session regroupe des communications qui traitent de connaissances et d'ontologies. Le premier article étudie comment étendre les possibilités du RàPC grâce à des traces d'utilisation (A. Cordier, B. Mascret, A. Mille). A. Abou Assali, D. Lenne et B. Debray étudient une similarité structurelle exploitant une ontologie dans une application de diagnostic. J. Cojan et J. Lieber traitent ensuite la combinaison de cas en s'appuyant sur la fusion de bases de connaissances. Des éléments de réflexion sur l'apport du RàPC dans la réutilisation de patrons d'ontologies et composants d'ontologies sont ensuite proposés (B. Fuchs et A. Napoli).
- Dans cette cinquième session, Vincent Duquenne (Université Pierre et Marie Curie, Paris 6), un invité d'une communauté proche, vient de l'analyse des données et de l'analyse formelle de concepts. Il nous fera profiter de son expérience en matière d'analyse de données complexes et de construction de treillis de concepts, ce qui intéresse plus d'une personne du monde du RàPC.
- La sixième session est divisée en deux parties.
  Dans la session « Applications », K. Haouchine, B. Chebel-Morello et N. Zerhouni proposent une mise en oeuvre du RàPC dans les domaines du diagnostic industriel tandis que C. Bobineau et L. Martinez-Medina utilisent le RàPC pour l'optimisation de requêtes.
  La session « Au tour de Taaable (2) » regroupe deux autres propositions issues du projet Taaable. La

première porte sur une algèbre temporelle permettant de représenter et manipuler par un raisonnement temporel les aspects temporels contenus dans des recettes de cuisine (F. Le Ber, J. Lieber, A. Napoli) et la deuxième traite d'évaluation des connaissances (S. Despres, H. Zargayouna, R. Bentebibel).

Les deux responsables scientifiques de cet atelier voudraient remercier toutes les personnes qui ont contribué à mettre en place cet atelier. Il y a les auteurs bien sûr, les lecteurs dont le travail de relecture est important et qui a été réalisé de façon soignée et cela dans des délais très courts. Il y a encore le conférencier invité et les chercheurs en analogie qui vont tous fortement contribuer à faire l'originalité de cet atelier. Enfin il y a le Laboratoire de Santé Publique et Informatique Médicale (SPIM) de Paris qui nous accueille et met à notre disposition locaux et moyens sous la houlette de Marie-Christine Jaulent et Rim Bentebibel.

Un dernier point : merci au LORIA et au LIRIS qui ont apporté leur aide financière à l'organisation du séminaire.

Béatrice Fuchs et Amedeo Napoli

TABLE DES MATIÈRES

COMITÉ DE PROGRAMME

- Fadi Badra, LORIA, Nancy
- Romain Bénard, LISyC, Brest
- Rim Bentebibel, LIPN, Paris
- Brigitte Chebel Morello, LAB, Basançon
- Amélie Cordier, LIRIS, Lyon
- Mathieu d'Aquin, KMi, Open University,
- Pierre De Loor, LISyC, Brest
- Sylvie Després, LIPN, Paris
- Béatrice Fuchs, LIRIS, Lyon
- Marie-Christine Jaulent, SPIM, Paris
- Rushed Kanawati, LIPN, Paris
- Florence Le Ber, LORIA, Nancy
- Jean Lieber, LORIA, Nancy
- Sophie Loriette-Rougegrez, UTT, Troyes
- Alain Mille, LIRIS, Lyon
- Amedeo Napoli, LORIA, Nancy
- Jean Renaud, ERPI, Nancy
- Karim Sehaba, LIRIS, Lyon
- Sylvie Salotti, LIPN, Paris
- Brigitte Trousse, INRIA, Sophia Antipolis

Some applications of lattice analysis for modeling concept systems

Vincent Duquenne

CNRS FRE 3232 & Univ. Paris 6,
175 rue du Chevaleret F-75013 Paris,
duquenne@math.jussieu.fr

Abstract.

The *Galois (or concept) Lattice* of a *binary relation* formalizes it as a *concept system*, dually ordered in "*extension*" / "*intension*". All implications between conjunctions of properties holding in it are summarized by a *canonical basis* -all basis having the same cardinality. These tools and lattices have been used to analyze binary data since the early eighties. Some procedures address more the former direction (*lattice drawing*, *gluing decomposition*, *sublattices* ...) while others the latter (*canonical basis of implications* and *lattice's core*, *box decomposition*, *weighted lattices*...).

More than entering into technicalities (for underlying theorems see [1] and for algorithms or methodological positions see [2]) we will herein address several examples coming from observational disciplines to give some hints on what can be gained through *lattice analysis* for modeling and deciphering concept systems.

[1] Ganter B. and R. Wille, *Formal concept analysis: mathematical foundations* (Springer, Berlin 1999).

[2] Duquenne V., Latticial structures in Data Analysis, *Theoretical Computer Science 217* (1999) 407-436 [download at **http://www.ecp6.jussieu.fr/pageperso/duquenne/duquenne.html**].

Acquisition et extraction de connaissances pour le système de raisonnement à partir de cas WIKITAAABLE

Fadi Badra[1], Julien Cojan[1], Amélie Cordier[2], Jean Lieber[1],
Thomas Meilender[1], Alain Mille[2], Pascal Molli[1], Emmanuel Nauer[1],
Amedeo Napoli[1], Hala Skaf-Molli[1], Yannick Toussaint[1]
[1]LORIA UMR 7503 CNRS, INRIA, Université de Nancy BP 239
54506 Vandœuvre-lès-Nancy, France, Prénom.Nom@loria.fr
[2]LIRIS CNRS, UMR 5202, Université Lyon 1, INSA Lyon, Université Lyon 2, ECL
43, bd du 11 Novembre 1918, Villeurbanne Cedex, France, Prénom.Nom@liris.cnrs.fr

Cet article a également été accepté à l'atelier du Computer Cooking Contest *de 2009, sous le titre* Knowledge Acquisition and Discovery for the Textual Case-Based Cooking system WIKITAAABLE .

Abstract

The textual case-based cooking system WIKITAAABLE participates to the second Computer cooking contest (CCC). It is an extension of the TAAABLE system that has participated to the first CCC. WIKITAAABLE's architecture is composed of a semantic wiki used for the collaborative acquisition of knowledge (recipe, ontology, adaptation knowledge) and of a case-based inference engine using this knowledge for retrieving and adapting recipes. This architecture allows various modes of knowledge acquisition for case-based reasoning that are studied within the TAAABLE project. In particular, opportunistic adaptation knowledge discovery is an approach for interactive and semi-automatic learning of adaptation knowledge triggered by a feedback from the user.

Keywords: textual case-based reasoning, case-based cooking, semantic wiki, opportunistic knowledge acquisition

URL of the system: `http://taaable.fr` is the URL of a Web page with a hyperlink on the WIKITAAABLE system (and also a hyperlink on the TAAABLE system that has participated to the first CCC).

1 Introduction

As final result from last year did not make us good cooks, we decided to keep on doing research. Hence, for the second edition of the Computer Cooking Contest (CCC), the TAAABLE system has evolved towards a new architecture called WIKITAAABLE. This year, we focused our efforts on two intertwined aspects: knowledge and reasoning. Concerning reasoning, we worked on the inference engine improvement to enhance the adaptation ability of the system. Concerning knowledge we set up advanced knowledge acquisition facilities to support the evolution of the system during its life-cycle.

This paper describes the innovations developed in WIKITAAABLE. This system presents various aspects that are detailed in other papers of the same proceedings. The aim of this paper is to give a general idea of the system and of the links between the system components. The WIKITAAABLE architecture is described in section 2. One innovation this year is that the system is embedded within a semantic wiki and that the collaborative aspects are also of main concern mainly for document and knowledge edition and update. The remainder of the paper shows how knowledge is manipulated within the system. Section 3 presents knowledge acquisition and representation within the semantic wiki. Section 4 illustrates how knowledge is used by the inference engine. Section 5 describes an opportunistic acquisition strategy guiding the evolution of the system knowledge. Finally, section 6 draws some conclusions and points out ongoing and future work. For qualification

purpose, a restricted interface of the system is available online. This interface only allows users to ask queries to the system. The full application with embedded knowledge acquisition features will be available for the contest.

2 Architecture of WIKITAAABLE

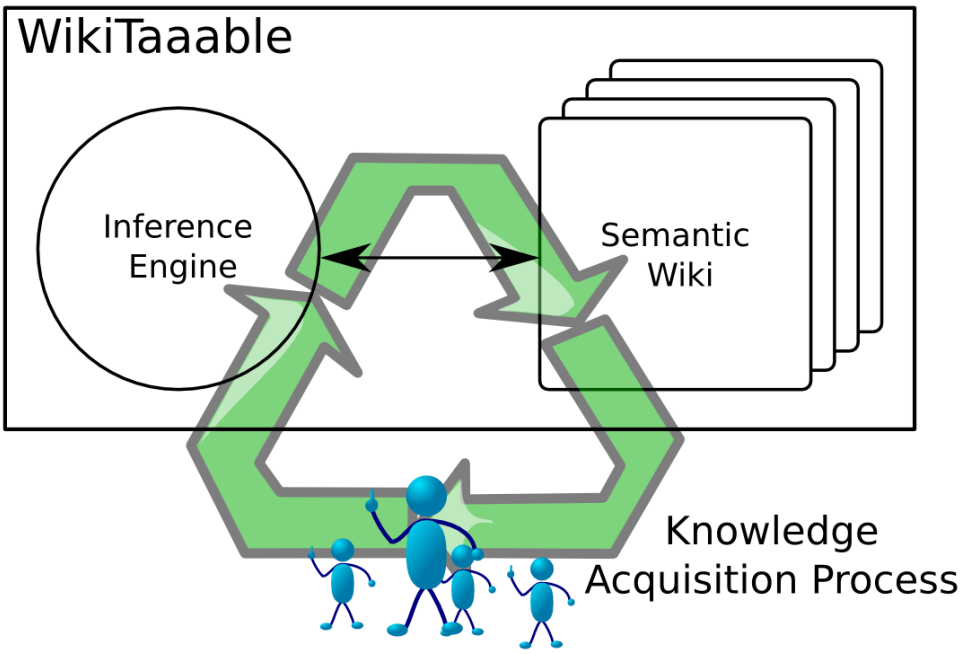


Figure 1: Overview of the WIKITAAABLE architecture.

In a CBR system, results strongly depend on the quality of available knowledge. As a consequence, continuous improvement of knowledge is required to progressively enhance the obtained results.

The previous version of TAAABLE [1] suffered from different problems making maintenance and evolution of the system knowledge difficult. For example, there was no way to capture user feedback and to reuse it for maintenance. Besides, knowledge was organized in several files of heterogeneous formats that were difficult to update. As a consequence, the evolutions of the system knowledge were tedious tasks, even for the designers of TAAABLE.

In WIKITAAABLE [3] we decided to use a semantic wiki (Semantic Media Wiki [7]) as a central module to manage all data and knowledge used in the system. Making use of a semantic wiki has two major advantages: it enables humans and machines to rely on the same tool for representing and reasoning on shared knowledge and it provides users with user-friendly interfaces for browsing and editing knowledge.

Figure 1 gives an overview of the WIKITAAABLE architecture. Each component has been designed to work with the others and the components are strongly intertwined. For example, knowledge has not been represented at a general level but for reasoning purpose. The *semantic wiki* module manages knowledge of the system. The wiki is accessible for the users through a graphical interface and for the system through bots[1] that automates several tasks. Section 3 details this module. The *inference engine* includes the CBR core and is able to reason on the knowledge available in the wiki. It is described in section 4. In order to facilitate knowledge acquisition, the architecture of WIKITAAABLE is designed in such a way that it enables as much interactions as possible. *Opportunistic knowledge acquisition process* developed in WIKITAAABLE is discussed in section 5.

[1] A bot is a piece of software for doing automated repetitive tasks.

3 A Semantic Wiki for Collaborative Acquisition of Cooking Knowledge

In [3], Semantic Media Wiki (SMW [7]) is used as a blackboard for WIKITAAABLE knowledge management. WIKITAAABLE gathers the whole knowledge body required by the application.

To import knowledge of the first version of the TAAABLE system [1] into WIKITAAABLE, we wrote several bots that use mediawiki API. Recipes, ontologies, and specific adaptation knowledge are now represented as a graph of semantic wiki pages. Each page can be freely read an updated by humans and by bots. Hence, TAAABLE is now maintained and improved by a collaboration between users and machines.

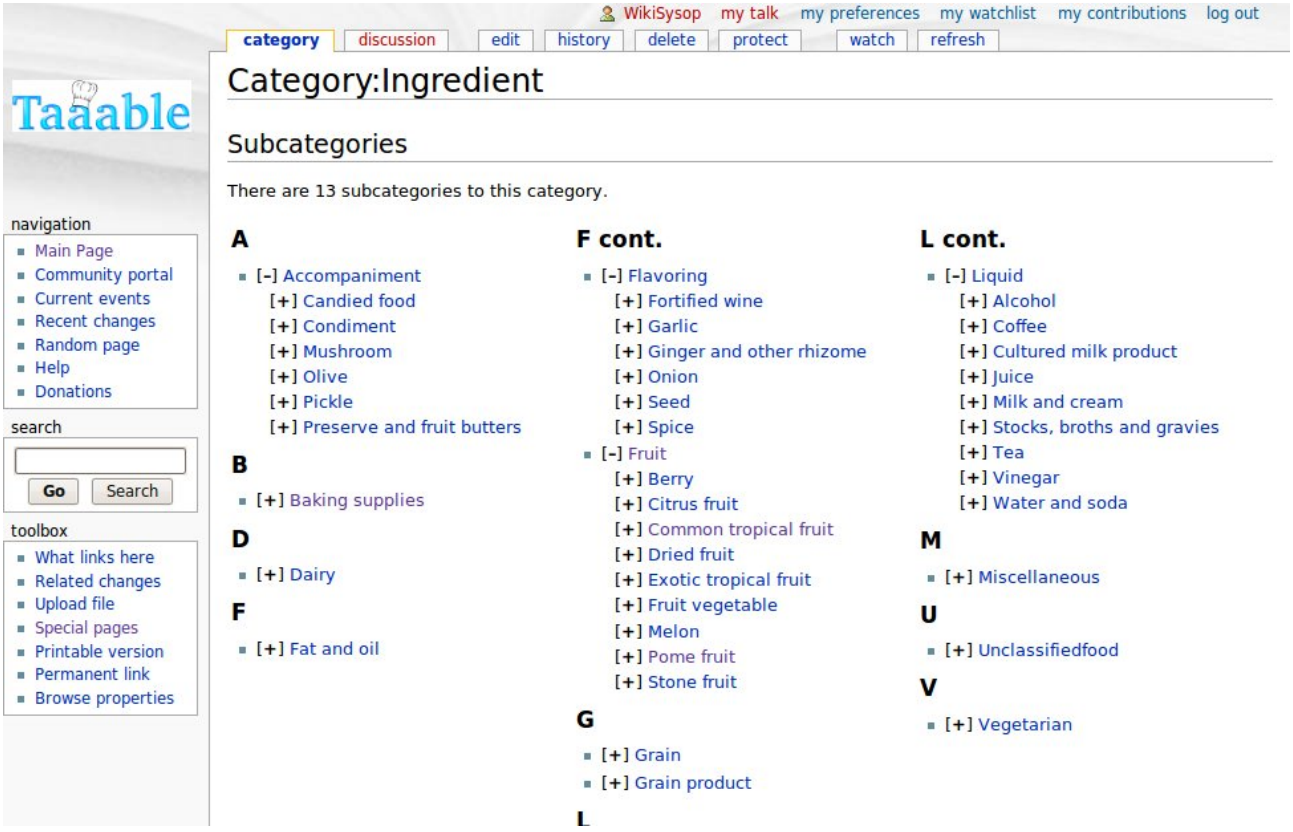


Figure 2: WIKITAAABLE ingredient ontology.

3.1 Domain Ontology

The domain ontology contains four hierarchies: *ingredients, dish moments, dish types, and dish origins*. For adapting a recipe by substituting some ingredients by other ingredients, the CBR engine requires knowledge stating similarity between ingredients. Therefore, ingredients are organized in the *ingredient hierarchy*. This hierarchy is used by the CBR engine to compute the cost of a substitution: the closer the ingredients, the lower the cost; e.g., *orange* is closer to *lemon* than *apple*.[2] In order to characterize recipes, three other hierarchies define and organize dish moments (*appetizer*, *dessert*), dish types (*cake*, *pizza*), and dish origins (*Mediterranean*, *Chinese*). The original acquisition of the hierarchies is described in [1].

The four hierarchies are imported into WIKITAAABLE by using the *Category/Sub-Category* relation of Semantic MediaWiki [7]. For example, there is a page for orange and another page for citrus and the two pages are linked by this relation. For instance, the figure 2 shows the imported ingredient hierarchy.

[2]This closeness can be measured by a weighted length of the shortest path between ingredients in the hierarchy.

3.2 Adaptation knowledge

To adapt a recipe, the CBR engine uses the ontology and a set AK of substitutions. A substitution $\sigma \in$ AK states that, in a given context, some ingredients may be substituted by other ones. For instance, the following substitution states that, for the context of a salad without potato, vinegar may be substituted by lemon juice and salt.

$$\sigma = \begin{array}{|ll|ll|ll|} \hline \textbf{context} & \text{salad} & \textbf{from} & \text{vinegar} & \textbf{by} & \text{lemon juice} \\ & \textbf{no} \text{ potato} & & & & \text{salt} \\ \hline \end{array} \quad (1)$$

Each substitution is represented in a semantic wiki page. For instance, figure 3 shows the wiki page of the above substitution. The acquisition of substitutions is detailed in section 5.

Figure 3: A substitution semantic wiki page.

3.3 Recipes

The recipes are also imported into WIKITAAABLE, where a wiki page is created for each recipe. Then, a bot crawls all the recipe pages, parses ingredient information, sets dish types, moments, and origins. It updates recipe pages with this information encoded as semantic annotations. Figure 4 shows a recipe page in WIKITAAABLE. We used the n-ary relationship of Semantic Media Wiki to represent an ingredient line, for example, $(1, \text{c}, \texttt{Category:rice})$ represents `1 c rice`, the first ingredient line in figure 4. The parsing of ingredient information and setting types is described in [1].

4 Principles of the CBR Inference

This section presents the main aspects of knowledge representation and of inferences in the WIKITAAABLE engine. They are presented with more details in [5].

4.1 Knowledge Containers

Knowledge in WIKITAAABLE is mainly expressed in propositional logic. The TAAABLE knowledge base is a set of containers $\texttt{KB} = \{\mathcal{O}, \texttt{Recipes}, \mathcal{H}_{idx}, \texttt{AK}, \texttt{cost}\}$. KB is encoded in wiki pages and the CBR engine has access to these pages through SPARQL queries.

$\mathcal{O}$ is the domain ontology represented by a set of axioms of the form $a \Rightarrow b$ where a and b are two variables representing recipe classes. For example, `lemon` (resp., `citrus`) represents the class of recipes with lemon (resp., with citrus) and the axiom $\texttt{lemon} \Rightarrow \texttt{citrus}$ states that any recipe with lemon is a recipe with citrus. In fact, every ingredient name X such as `lemon` is interpreted here as "class of the recipes with ingredient X".

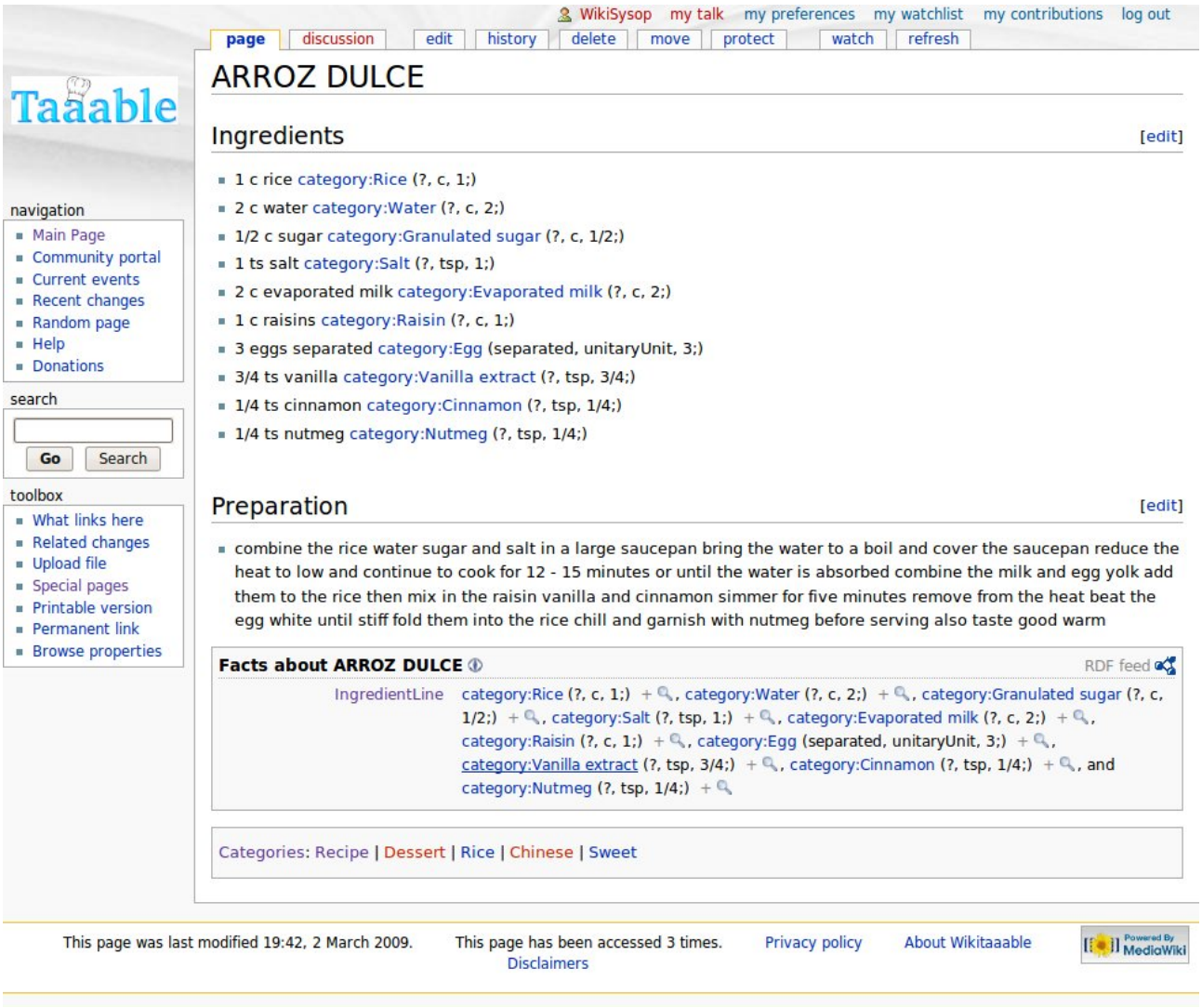


Figure 4: Indexed recipe of “Arroz dulce”.

`Recipes` is the set of recipes given by the CCC organizers, and consequently the case base of the CBR system TAAABLE. A recipe $R \in$ `Recipes` cannot be directly handled by the CBR inference engine: the engine requires a formal representation whereas R is for the largest part written in natural language. Therefore, only the formalized part of the recipe R is used: its index $idx(R)$, which is expressed by a conjunction of literals (the indexing process of the recipes coincides with the annotation process mentioned in section 3.3). For example

$$idx(R) = \texttt{lettuce} \wedge \texttt{vinegar} \wedge \texttt{olive\_oil} \wedge \texttt{tomato} \wedge \textit{Nothing else} \qquad (2)$$

is a formal representation of a recipe R having ingredients lettuce, vinegar, olive oil, and tomato. In this expression, “*Nothing else*” indicates that a closed world assumption is associated to $idx(R)$ stating that if a property cannot be deduced from the recipe description and from the ontology then it is considered as false. Formally, if $idx(R) \not\models_{\mathcal{O}} a$ then “*Nothing else*” contains the conjunct $\neg a$.[3] For example, this closed world assumption enables to deduce that $idx(R) \models_{\mathcal{O}} \neg\texttt{meat} \wedge \neg\texttt{fish}$, i.e., that R is a vegetarian recipe.

The indexes $idx(R)$ are used to access recipes through a hierarchy $\mathcal{H}_{idx}$, according to the partial ordering $\models_{\mathcal{O}}$: for $C, D \in \mathcal{H}_{idx}$, $C \models_{\mathcal{O}} D$ iff there is a directed path in $\mathcal{H}_{idx}$ from C to D. The indexes $idx(R)$ are leaves of the $\mathcal{H}_{idx}$.

[3] If f and g are two propositional formulas, $f \models_{\mathcal{O}} g$ means that f implies g, given the ontology $\mathcal{O}$. More precisely: if $\mathcal{I}$ satisfies both $\mathcal{O}$ and f then $\mathcal{I}$ satisfies g.

Adaptation knowledge has two parts. The first part is included in ontology $\mathcal{O}$. The second part is the set AK of substitutions (cf. section 3.2). Any $\sigma \in$ AK may be considered as a domain specific inference rule $\frac{R \text{ is a good recipe}}{\sigma(R) \text{ is a good recipe}}$. The substitutions have the form $C \rightsquigarrow D$ where C and D are conjunctions of literals. Applying $C \rightsquigarrow D$ to a conjunction of literals (such as an index or a query) consists in replacing the literals of C by literals of D. For example, the substitution σ described in figure 3 is written as follows

$$\sigma = \texttt{salad} \wedge \neg\texttt{potato} \wedge \texttt{vinegar} \rightsquigarrow \texttt{salad} \wedge \neg\texttt{potato} \wedge \texttt{lemon\_juice} \wedge \texttt{salt} \tag{3}$$

It can be noticed that the substitution given by a triple $(\texttt{context}, \texttt{from}, \texttt{by})$ in the wiki pages (cf. equation (1)) are rewritten $\texttt{context} \wedge \texttt{from} \rightsquigarrow \texttt{context} \wedge \texttt{by}$ to suit the CBR engine formalism.

The CBR inference is based on substitutions, either taken from AK or built with the help of ontology $\mathcal{O}$ (see below for details). The choice of substitutions is made according to the problem-solving context and to a cost function $\texttt{cost} : \sigma \mapsto \texttt{cost}(\sigma) > 0$; substitution σ is preferred to substitution τ when $\texttt{cost}(\sigma) < \texttt{cost}(\tau)$. Therefore, the cost function (and its parameters) constitutes an additional knowledge container.

4.2 CBR Inference

Let Q be a query. For example

$$Q = \texttt{endive} \wedge \texttt{lemon\_juice} \wedge \neg\texttt{onion} \tag{4}$$

is a query for a recipe with endive and lemon juice and without onion. The CBR inference consists in (1) retrieving recipes matching exactly or approximately Q and (2) adapting the retrieved recipes.

Retrieval aims at choosing indexes $idx(R)$ matching the query Q. An exact match corresponds to $idx(R) \vDash_{\mathcal{O}} Q$. If no index exactly matches Q, the query Q is progressively relaxed into $\Gamma(Q)$ such that $idx(R) \vDash_{\mathcal{O}} \Gamma(Q)$, for some $idx(R)$. The relaxation of Q is obtained by applying generalizations g_k according to $\mathcal{O}$: $g_k = a_k \rightsquigarrow b_k$ is a substitution such that $(a_k \Rightarrow b_k) \in \mathcal{O}$. Thus $\Gamma(Q) = g_n(\ldots(g_1(Q))\ldots)$. Therefore, retrieval provides a similarity path

$$idx(R) \vDash_{\mathcal{O}} \Gamma(Q) \xleftarrow{g_n} \ldots \xleftarrow{g_1} Q \tag{5}$$

This similarity path is built according to a best-first search minimizing $\sum_k \texttt{cost}(g_k)$. For example, retrieval can give the following result

$$\begin{aligned} Q &= \texttt{endive} \wedge \texttt{lemon\_juice} \wedge \neg\texttt{onion} \\ \Gamma(Q) &= \texttt{green\_salad} \wedge \top \wedge \neg\texttt{onion} \equiv \texttt{green\_salad} \wedge \neg\texttt{onion} \\ idx(R) &= \texttt{lettuce} \wedge \texttt{vinegar} \wedge \texttt{olive\_oil} \wedge \texttt{tomato} \wedge \textit{Nothing else} \end{aligned}$$

(Γ consists in generalizing endive into green_salad and in removing lemon_juice from the query by generalizing it in several steps to $\top$, the hierarchy top).

Adaptation is composed of two sub-steps. Let R be a retrieved recipe, with index $idx(R)$. The first subset of adaptation is matching, that aims at building an adaptation path from $idx(R)$ to Q of the form

$$idx(R) \xmapsto{\sigma_1} \ldots \xmapsto{\sigma_p} \Sigma(idx(R)) \vDash_{\mathcal{O}} \Gamma(Q) \xleftarrow{\gamma_q} \ldots \xleftarrow{\gamma_1} Q \tag{6}$$

where $\sigma_i \in$ AK $(i = 1 \ldots p)$ and substitutions γ_j $(j = 1 \ldots q)$ correspond to axioms of the ontology: $\gamma_j = a_j \rightsquigarrow b_j$ with $(a_j \Rightarrow b_j) \in \mathcal{O}$. Such an adaptation path is built according to a best-first search in a state space minimizing $\left(\sum_i \texttt{cost}(\sigma_i) + \sum_j \texttt{cost}(\gamma_j)\right)$.

The second sub-step of adaptation consists in "following" the adaptation path: first R is adapted successively in $\sigma_1(R)$, $\sigma_2(\sigma_1(R))$, $\ldots\sigma_p(\ldots(\sigma_2(\sigma_1(R))\ldots) = \Sigma(R)$. Then, ingredients of $\Sigma(R)$ are substituted by other ingredients according to a generalization-specialization process (generalization corresponds to the relation $\vDash_{\mathcal{O}}$ and specialization to the substitutions $\gamma_q^{-1}, \ldots, \gamma_1^{-1}$).

For example, let $idx(R)$ and Q be the example presented above. Matching may provide the following adaptation path:

$$idx(R) \overset{\sigma}{\longmapsto} \Sigma(idx(R)) \models_{\mathcal{O}} \Gamma(Q) \overset{\gamma}{\longleftarrow} Q$$

where σ is defined by (3) and $\gamma = \texttt{endive} \rightsquigarrow \texttt{green\_salad}$. Thus, the adaptation of R consists in replacing vinegar by lemon juice and salt (cf. σ) and by substituting `lettuce` by `endive` (cf. $\models_{\mathcal{O}}$ and γ^{-1}).

It can be noticed that retrieval provides a first matching: a similarity path is a kind of adaptation path involving no $\sigma \in \texttt{AK}$. Thus, the retrieved recipe R can be adapted following this similarity/adaptation path. Therefore, the retrieval process is adaptation-guided [6]: a retrieved recipe is ensured to be adaptable into a recipe matching query Q. However, during adaptation, some substitutions $\sigma \in \texttt{AK}$ may be used and, if they do, the resulting adaptation requires less effort.[4]

5 Opportunistic Knowledge Acquisition and Discovery

WIKITAAABLE has been designed to facilitate continuous knowledge acquisition through interactions with its users: it is a reflexive and perpetually evolving system. However, due to the heterogeneity of knowledge acquisition situations that can be envisioned, setting up such a process is a complex task. This diversity of situations is explained by several factors:

- The various types of knowledge (ontology, adaptation knowledge, substitutions costs, recipes) that can be acquired.
- The different interaction modalities such as simple user feedback, direct modification on wiki pages, interaction through dedicated interfaces, use of external knowledge discovery methods, etc.
- The provenance of knowledge that is acquired: single users, community of users, experts, other sources of data (web sites), or local knowledge from which new knowledge in inferred.

In the following, a particular scenario of opportunistic knowledge discovery is detailed. In WIKITAAABLE, substitutions $\sigma \in \texttt{AK}$ are acquired at problem-solving time through interactions with the user. The knowledge discovery process CABAMAKA [4] is used to assist the user in the formulation of new pieces of knowledge. Its role is to generate a set of substitutions $\sigma \in \texttt{AK}$ "on the fly" from the comparison of two sets of recipes of the case base. The generated substitutions can be used by the system to repair a failed adaptation. Each time a substitution is validated by the user, it is stored for future reuse. In the following, the main principles of the approach are illustrated on an example. More details on the proposed approach can be found in [2].

In section 4, the user wanted a salad recipe with lemon juice but without onion, which was modeled by the query Q defined by (4). The substitution $\sigma \in \texttt{AK}$ defined by (3) was used to adapt the retrieved recipe R by replacing vinegar by lemon juice and salt. Such a substitution cannot be obtained from the ontology $\mathcal{O}$, so let us assume in this scenario that σ is not available to the system. Thus, to perform adaptation, the system relies solely on the ontology $\mathcal{O}$ from which it generates the substitution $\texttt{vinegar} \rightsquigarrow \texttt{lemon\_juice}$. Now, in our scenario, the user is not satisfied with the proposed solution and gives this feedback to the system. Therefore, the knowledge discovery process is triggered: a set of substitutions is learned from the case base by comparing salad recipes with vinegar and salad recipes with lemon juice. Among the learned substitutions is the substitution $\sigma_{learned} = \texttt{vinegar} \rightsquigarrow \texttt{lemon\_juice} \wedge \texttt{salt}$, which suggests to replace vinegar by lemon juice in the retrieved recipe R and to add salt. The user is satisfied with the adaptation resulting from the application of this substitution, so the latter is stored for future reuse. At this point, the user is encouraged to specify the condition of application of the substitution $\sigma_{learned}$. The user states that vinegar can be replaced by lemon juice and salt in salad recipes that do not contain potato, which is modeled by the substitution context $\texttt{salad} \wedge \neg \texttt{potato}$. Combining the learned substitution $\sigma_{learned}$ and its application context gives the substitution σ defined by (3).

[4] If the cost function is an estimation of the adaptation effort, then the adapted recipe should be better by following (6) then by following (5). Indeed, since adding new substitutions (the ones of AK) only adds new ways to connect indexes to queries, it comes that $\sum_i \texttt{cost}(\sigma_i) + \sum_j \texttt{cost}(\gamma_j) \leq \sum_k \texttt{cost}(g_k)$.

In WIKITAAABLE, the wiki is used as a gateway enabling to centralize knowledge used in the system. It provides functionalities to facilitate acquisition and maintenance of knowledge and enables to progressively add new acquisition features, allowing the evolution of the whole system. However, setting up a complex knowledge acquisition process raises several issues. For example, tools for ensuring consistency of knowledge used in the system must be developed. Another issue is to handle updates from multiple users. What happens when one believes that an avocado is to be eaten as a starter whereas someone else reckon that it has to be eaten as a dessert? Is the system supposed to converge towards a "commonly accepted" knowledge base or should it be able to deal with user's preferences?

A strength of the architecture of WIKITAAABLE is that it will enable to progressively address these issues. A future work is to allow users to add their own recipes to the system. This functionality requires to be able to dynamically annotate new recipes within WIKITAAABLE in order to make them usable by the CBR inference engine. One of the advantages of such a functionality, combined with the benefits of a wiki, is that communities of users will be able to share their recipes and to collaborate in order to improve the global knowledge of the system. Next, we would like to tackle the multi-user issue which is a prerequisite for envisioning a collaborative building of the knowledge base of TAAABLE.

6 Conclusion, Ongoing Work, and Future Work

The textual case-based cooking system WIKITAAABLE participates to the second CCC. It is an extension of the TAAABLE system that has participated to the first CCC. WIKITAAABLE's architecture is composed of a semantic wiki used for the collaborative acquisition of knowledge (recipe, ontology, adaptation knowledge) and of a CBR inference engine using this knowledge for retrieving and adapting recipes. This architecture allows various modes of knowledge acquisition for CBR that are studied within the TAAABLE project. In particular, opportunistic adaptation knowledge discovery is an approach for interactive and semi-automatic learning of adaptation knowledge triggered by a feedback from the users.

The first ongoing work is the improvement of the WIKITAAABLE system (user interface, inference engine, knowledge base encoded in wiki pages, and links between these components). Another work planned for the next weeks is the development of tools within WIKITAAABLE for knowledge acquisition triggered by user feedbacks. Such a knowledge acquisition leads to a continuous evolution of the knowledge base and thus, of the behavior of the system. It is important to ensure that these evolutions are improvements and that the integrity of the knowledge is preserved. We plan to use non regression and consistency tests for this purpose.

Currently, wiki pages are accessed and maintained by a limited community: the TAAABLE project members. These pages encode the knowledge that have been acquired on the basis of a consensus. A long term objective is to have several open semantic wikis with cooking knowledge, each of them corresponding to a user community, the consensus being only realized at the level of a community.

Acknowledgments

The participants of the TAAABLE project wish to thank the organizers of the CCC for providing this benchmark, that entails many interesting problems, and the need to collaborate with researchers in various domains on knowledge engineering. They also wish to thank the reviewers of this paper for their interesting remarks and suggestions. Some of these suggestions have not yet been taken into account in this paper since they involve long term research, but they raise interesting issues for the authors.

References

[1] Badra (F.), Bendaoud (R.), Bentebitel (R.), Champin (P.-A.), Cojan (J.), Cordier (A.), Desprès (S.), Jean-Daubias (S.), Lieber (J.), Meilender (T.), Mille (A.), Nauer (E.), Napoli (A.) et Toussaint (Y.). – Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. *In : ECCBR Workshops, Workshop of the First Computer Cooking Contest*, pp. 219–228. – 2008.

[2] Badra (F.), Cordier (A.) et Lieber (J.). – Découverte opportuniste de connaissances d'adaptation. *In : À paraître dans les actes du 17ème atelier raisonnement à partir de cas (RàPC-09)*. – 2009.

[3] Cordier (A.), Lieber (J.), Molli (P.), Nauer (E.), Skaf-Molli (H.) et Toussaint (Y.). – Wikitaaable: A semantic wiki as a blackboard for a textual case-based reasoning system. *In : SemWiki 2009 – 4th Semantic Wiki Workshop*. – Heraklion, Greece, May 2009.

[4] d'Aquin (M.), Badra (F.), Lafrogne (S.), Lieber (J.), Napoli (A.) et Szathmary (L.). – Case base mining for adaptation knowledge acquisition. *In : Proceedings of the International Conference on Artificial Intelligence, IJCAI'07*, pp. 750–756. – 2007.

[5] Lieber (J.). – Le moteur de raisonnement à partir de cas de WIKITAAABLE. *In : À paraître dans les actes du 17ème atelier raisonnement à partir de cas (RàPC-09)*. – 2009.

[6] Smyth (B.) et Keane (M. T.). – Using adaptation knowledge to retrieve and adapt design cases. *Knowledge-Based Systems*, vol. 9, n2, 1996, pp. 127–135.

[7] Völkel (M.), Krötzsch (M.), Vrandecic (D.), Haller (H.) et Studer (R.). – Semantic Wikipedia. *Journal of Web Semantics*, vol. 5, n4, 2007.

Découverte opportuniste de connaissances d'adaptation

Fadi Badra[1], Amélie Cordier[2], Jean Lieber[1]
[1] LORIA (CNRS, INRIA, Nancy Universities)
BP 239, 54506 Vandœuvre-lès-Nancy, France
{badra,lieber}@loria.fr
[2] LIRIS CNRS UMR 5202, Université Lyon 1, INSA Lyon, Université Lyon 2, ECL
43, bd du 11 novembre 1918, Villeurbanne, France
Amelie.Cordier@liris.cnrs.fr

Résumé

L'étape d'adaptation est souvent considérée comme le talon d'Achille du raisonnement à partir de cas car elle nécessite des connaissances spécifiques au domaine d'application qui sont difficiles à acquérir. Dans ce papier, deux stratégies sont combinées pour faciliter la tâche d'acquisition de connaissances d'adaptation : les connaissances d'adaptation sont apprises à partir de la base de cas par des techniques d'extraction de connaissances, et l'acquisition de connaissances d'adaptation est déclenchée de manière opportuniste au cours d'une session particulière de résolution de problèmes.

Mots clés : Acquisition de connaissances d'adaptation, Extraction de connaissances

1 Introduction

Case-based reasoning (CBR [8]) is a reasoning paradigm based on the reuse of previous problem-solving experiences, called cases. A CBR system often has profit of a retrieval procedure, selecting in a case base a source case similar to the target problem, and an adaptation procedure, that adapts the retrieved source case to the specificity of the target problem. The adaptation procedure depends on domain-dependent adaptation knowledge (AK, in the following). Acquiring AK can be done from experts or by using machine learning techniques. An intermediate approach is knowledge discovery (KD) that combines efficient learning algorithms with human-machine interaction.

Most of previous AK acquisition strategies are off-line : they are disconnected from the use of the CBR system. By contrast, recent work aims at integrating AK acquisition from experts to specific reasoning sessions : this *opportunistic* AK acquisition takes advantage of the problem-solving context. This paper presents an approach to AK discovery that is opportunistic : the KD is triggered at problem-solving time.

The paper is organized as follows. Section 2 introduces some basic notions and notations about CBR. Section 3 presents the CBR system Taaable, which constitutes the application context of the study, and motivates the need for adaptation knowledge acquisition in this application context. Section 4 presents the proposed opportunistic and interactive AK discovery method. In Sect. 5, this method is applied to acquire adaptation knowledge in the context of the Taaable system. Section 6 discusses this approach and situates it among related work. Section 7 concludes and presents some future work.

2 Basic Notions About CBR

In the following, problems are assumed to be represented in a language $\mathcal{L}_{\mathtt{pb}}$ and solutions in a language $\mathcal{L}_{\mathtt{sol}}$. A *source case* represents a problem-solving episode by a pair (`srce`, `Sol`(`srce`)), in which

$\texttt{srce} \in \mathcal{L}_{\texttt{pb}}$ is the representation of a problem statement and $\texttt{Sol}(\texttt{srce}) \in \mathcal{L}_{\texttt{sol}}$ is the representation of its associated solution. CBR aims at solving a *target problem* tgt using a set of source cases CB called the *case base*. The CBR process is usually decomposed in two main steps : retrieval and adaptation. *Retrieval* selects a source case $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ from the case base such that srce is judged to be similar to tgt according to a given similarity criterion. *Adaptation* consists in modifying $\texttt{Sol}(\texttt{srce})$ in order to propose a candidate solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ for tgt to the user. If the user validates the candidate solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$, then $\widetilde{\texttt{Sol}}(\texttt{tgt})$ is considered to be a solution $\texttt{Sol}(\texttt{tgt})$ for tgt.

3 Application Context : the TAAABLE System

The TAAABLE system [2] is a cooking CBR system. In the cooking domain, CBR aims at answering a query using a set of recipes. In order to answer a query, the system retrieves a recipe in the recipe set and adapts it to produce a recipe satisfying the query. The TAAABLE system was proposed to participate to the *Computer Cooking Contest* (CCC) challenge in 2008 [14]. In the CCC challenge, queries are given in natural language and express a set of constraints that the desired recipe should satisfy. These constraints concern the ingredients to be included or avoided, the type of ingredients (e.g., meat or fruit), the dietary practice (e.g., nut-free diet), the type of meal (e.g., soup) or the type of cuisine (e.g., chinese cuisine). An example of query is : "Cook a chinese soup with leek but no peanut oil." Recipes are given in textual form, with a shallow XML structure, and include a set of ingredients together with a textual part describing the recipe preparation. The TAAABLE system is accessible online (http://taaable.fr).

3.1 Representation Issues

A Cooking Ontology. The system makes use of a cooking ontology $\mathcal{O}$ represented in propositional logic. Each concept of $\mathcal{O}$ corresponds to a propositional variable taken from a finite set $\mathcal{V}$ of propositional variables. $\mathcal{O}$ is mainly composed of a set of concepts organized in a hierarchy, which corresponds, in propositional logic, to a set of logical implications $\texttt{a} \Rightarrow \texttt{b}$. For example, the axiom $\texttt{leek} \Rightarrow \texttt{onions}$ of $\mathcal{O}$ states that leeks are onions.

Problem and Solution Representation. In TAAABLE, a problem $\texttt{pb} \in \mathcal{L}_{\texttt{pb}}$ represents a query and a solution $\texttt{Sol}(\texttt{pb})$ of pb represents a recipe that matches this query. $\mathcal{L}_{\texttt{pb}}$ and $\mathcal{L}_{\texttt{sol}}$ are chosen fragments of propositional logic defined using the vocabulary $\mathcal{V}$ introduced in the cooking ontology $\mathcal{O}$. One propositional variable is defined in $\mathcal{L}_{\texttt{pb}}$ and $\mathcal{L}_{\texttt{sol}}$ for each concept name of $\mathcal{O}$ and the only logical connective used in $\mathcal{L}_{\texttt{pb}}$ and $\mathcal{L}_{\texttt{sol}}$ is the conjunction $\wedge$. For example, the representation $\texttt{tgt} \in \mathcal{L}_{\texttt{pb}}$ of the query mentioned above is :

$$\texttt{tgt} = \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{leek} \wedge \neg\, \texttt{peanut\_oil}$$

The case base CB contains a set of recipes. Each recipe is indexed in the case base by a propositional formula $\texttt{R} \in \mathcal{L}_{\texttt{sol}}$. For example, the index R of the recipe *Wonton Soup* is :

$$\texttt{R} = \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{green\_onion} \wedge \ldots \wedge \texttt{peanut\_oil} \wedge \textit{Nothing else}$$

Nothing else denotes a conjunction of negative literals $\neg\, \texttt{a}$ for all $\texttt{a} \in \mathcal{V}$ such that $\texttt{chinese} \wedge \texttt{soup} \wedge \texttt{green\_onion} \wedge \ldots \wedge \texttt{peanut\_oil} \not\models_{\mathcal{O}} \texttt{a}$. This kind of "closed world assumption" states explicitly that for all propositional variable $\texttt{a} \in \mathcal{V}$, either $\texttt{R} \models_{\mathcal{O}} \texttt{a}$ (the recipe contains the ingredient represented by a) or $\texttt{R} \models_{\mathcal{O}} \neg\, \texttt{a}$ (the recipe does not contain the ingredient represented by a).

Each recipe index R represents a set of source cases : R represents the set of source cases $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ such that $\texttt{Sol}(\texttt{srce}) = \texttt{R}$ and srce is solved by R, i.e., srce is such that $\texttt{R} \models_{\mathcal{O}} \texttt{srce}$.

Adaptation Knowledge. In TAAABLE, adaptation knowledge is given by a set of reformulations $(\mathtt{r}, \mathcal{A}_\mathtt{r})$ in which $\mathtt{r}$ is a binary relation between problems and $\mathcal{A}_\mathtt{r}$ is an adaptation function associated with $\mathtt{r}$ [13]. A reformulation has the following semantics : if two problems $\mathtt{pb}_1$ and $\mathtt{pb}_2$ are related by $\mathtt{r}$ —denoted by $\mathtt{pb}_1 \ \mathtt{r} \ \mathtt{pb}_2$— then for every recipe $\mathtt{Sol}(\mathtt{pb}_1)$ matching the query $\mathtt{pb}_1$, $\mathcal{A}_\mathtt{r}(\mathtt{pb}_1, \mathtt{Sol}(\mathtt{pb}_1), \mathtt{pb}_2) = \widetilde{\mathtt{Sol}}(\mathtt{pb}_2)$ matches the query $\mathtt{pb}_2$.

In this paper, binary relations $\mathtt{r}$ are given by substitutions of the form $\sigma = \alpha \rightsquigarrow \beta$, where α and β are literals (either positive or negative). For example, the substitution $\sigma = \mathtt{leek} \rightsquigarrow \mathtt{onions}$ generalizes leek into onions.

Adaptation functions $\mathcal{A}_\mathtt{r}$ are given by substitutions of the form $\Sigma = \mathtt{A} \rightsquigarrow \mathtt{B}$ in which $\mathtt{A}$ and $\mathtt{B}$ are conjunctions of literals. For example, the substitution $\Sigma = \mathtt{soup} \wedge \mathtt{pepper} \rightsquigarrow \mathtt{soup} \wedge \mathtt{ginger}$ states that pepper can be replaced by ginger in soup recipes. A substitution Σ can be automatically generated from a substitution $\sigma : \Sigma = \mathtt{b} \rightsquigarrow \mathtt{a}$ if σ is of the form $\mathtt{a} \rightsquigarrow \mathtt{b}$ and $\Sigma = \emptyset \rightsquigarrow \neg\, \mathtt{a}$ if σ is of the form $\neg\, \mathtt{a} \rightsquigarrow \emptyset$.

The main source of adaptation knowledge is the ontology $\mathcal{O}$. A substitution $\sigma = \mathtt{a} \rightsquigarrow \mathtt{b}$ is automatically generated from each axiom $\mathtt{a} \Rightarrow \mathtt{b}$ of $\mathcal{O}$ and correspond to a *substitution by generalization*. A substitution $\sigma = \mathtt{a} \rightsquigarrow \mathtt{b}$ can be applied to a query $\mathtt{pb}$ if $\mathtt{pb} \models_\mathcal{O} \mathtt{a}$. σ generates a new query $\sigma(\mathtt{pb})$ in which the propositional variable $\mathtt{a}$ has been substituted by the propositional variable $\mathtt{b}$. For example, the substitution $\sigma = \mathtt{leek} \rightsquigarrow \mathtt{onions}$ is generated automatically from the axiom $\mathtt{leek} \Rightarrow \mathtt{onions}$ of $\mathcal{O}$. σ can be applied to the query $\mathtt{tgt}$ to produce the query $\sigma(\mathtt{tgt}) = \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{onions} \wedge \neg\, \mathtt{peanut\_oil}$, in which $\mathtt{leek}$ has been substituted by $\mathtt{onions}$. For each propositional variable $\mathtt{a}$ of $\mathcal{V}$, an additional substitution of the form $\sigma = \neg\, \mathtt{a} \rightsquigarrow \emptyset$ is generated. Such a substitution can be applied to a problem $\mathtt{pb}$ if $\mathtt{pb} \models_\mathcal{O} \neg\, \mathtt{a}$ and generates a new problem $\sigma(\mathtt{pb})$ in which the negative literal $\neg\, \mathtt{a}$ is removed. This has the effect to loosen the constraints imposed on a query e.g., by omitting in the query an unwanted ingredient. For example, the substitution $\neg\, \mathtt{peanut\_oil} \rightsquigarrow \emptyset$ applied to $\mathtt{tgt}$ generates the query $\sigma(\mathtt{tgt}) = \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{leek}$, in which the condition on the ingredient $\mathtt{peanut\_oil}$ is omitted.

However, when $\mathcal{O}$ is the only source of adaptation knowledge, the system is only able to perform simple adaptations, in which the modifications made to $\mathtt{Sol}(\mathtt{srce})$ correspond to a sequence of substitutions that can be used to transform $\mathtt{srce}$ into $\mathtt{tgt}$. Therefore, an additional adaptation knowledge base $\mathtt{AKB}$ is introduced. $\mathtt{AKB}$ contains a set of reformulations (σ, Σ) that capture more complex adaptation strategies.

3.2 The CBR Process in TAAABLE

This section presents a summary of the TAAABLE CBR process. A more detailed presentation can be found in [12].

Retrieval. The retrieval algorithm is based on a *smooth classification* algorithm on an index hierarchy. Such an algorithm aims at determining a set of modifications to apply to $\mathtt{tgt}$ in order to obtain a modified query $\mathtt{srce}$ that matches at least one recipe $\mathtt{Sol}(\mathtt{srce})$ of the case base. The algorithm computes a *similarity path*, which is a composition of substitutions $\mathtt{SP} = \sigma_q \circ \sigma_{q-1} \circ \cdots \circ \sigma_1$ such that there exists at least one recipe $\mathtt{Sol}(\mathtt{srce})$ matching the modified query $\mathtt{srce} = \sigma_q(\sigma_{q-1}(\ldots \sigma_1(\mathtt{tgt})\ldots))$, i.e., such that $\mathtt{Sol}(\mathtt{srce}) \models_\mathcal{O} \mathtt{srce}$ holds. Thus, a similarity path $\mathtt{SP}$ can be written :

$$\mathtt{Sol}(\mathtt{srce}) \models_\mathcal{O} \mathtt{srce} \xleftarrow{\sigma_q} \xleftarrow{\sigma_{q-1}} \cdots \xleftarrow{\sigma_1} \mathtt{tgt}$$

For example, to solve the above query $\mathtt{tgt}$, the system generates a similarity path $\mathtt{SP} = \sigma_2 \circ \sigma_1$, with :

$$\begin{aligned}
\mathtt{tgt} &= \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{leek} \wedge \neg\, \mathtt{peanut_oil} \\
\sigma_1 = \neg\, \mathtt{peanut_oil} \rightsquigarrow \emptyset, &\quad \sigma_2 = \mathtt{leek} \rightsquigarrow \mathtt{onions} \\
\mathtt{srce} &= \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{onions} \\
\mathtt{Sol}(\mathtt{srce}) &= \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{green_onion} \wedge \ldots \wedge \mathtt{peanut_oil} \wedge \textit{Nothing else}
\end{aligned}$$

In this similarity path, $\texttt{Sol}(\texttt{srce})$ is the propositional representation of the recipe *Wonton Soup*. Since the ontology O contains the axiom $\texttt{green\_onion} \Rightarrow \texttt{onions}$, the modified query $\texttt{srce} = \sigma_2 \circ \sigma_1(\texttt{tgt})$ verifies $\texttt{Sol}(\texttt{srce}) \models_O \texttt{srce}$.

Adaptation. To a similarity path is associated an *adaptation path* AP, which is a composition of substitutions $\texttt{AP} = \Sigma_1 \circ \Sigma_2 \circ \cdots \circ \Sigma_q$ such that the modified recipe $\widetilde{\texttt{Sol}}(\texttt{tgt}) = \Sigma_1(\Sigma_2(\ldots\Sigma_q(\texttt{Sol}(\texttt{srce}))\ldots))$ solves the initial query tgt, i.e., verifies $\widetilde{\texttt{Sol}}(\texttt{tgt}) \models_O \texttt{tgt}$. Thus, an adaptation path AP can be written

$$\texttt{Sol}(\texttt{srce}) \xrightarrow{\Sigma_q} \xrightarrow{\Sigma_{q-1}} \cdots \xrightarrow{\Sigma_1} \widetilde{\texttt{Sol}}(\texttt{tgt}) \models_O \texttt{tgt}$$

The adaptation path AP is constructed from the similarity path SP by associating a substitution Σ_i to each substitution σ_i. To determine which substitution Σ_i to associate to a given substitution σ_i, the external adaptation knowledge base AKB is searched first. For a substitution $\sigma_i = \alpha \rightsquigarrow \beta$, the system looks for a substitution $\Sigma = \texttt{A} \rightsquigarrow \texttt{B}$ such that $\texttt{A} \models_O \beta$ and $\texttt{B} \models_O \alpha$. For example, if $\sigma_2 = \texttt{leek} \rightsquigarrow \texttt{onions}$ is used in SP and AKB contains the reformulation (σ, Σ) with $\sigma = \sigma_2$ and $\Sigma = \texttt{green\_onion} \rightsquigarrow \texttt{leek} \wedge \texttt{ginger}$, Σ will be selected to constitute the substitution Σ_2 in AP since $\texttt{green\_onion} \models_O \texttt{onions}$ and $\texttt{leek} \wedge \texttt{ginger} \models_O \texttt{leek}$. If no substitution Σ is found in AKB for a given substitution σ_i then Σ_i is generated automatically from σ_i.

In the previous example, AKB is considered to be empty so Σ_1 and Σ_2 are generated automatically from the substitutions σ_1 and σ_2 : $\Sigma_1 = \emptyset \rightsquigarrow \neg\,\texttt{peanut\_oil}$ since $\sigma_1 = \neg\,\texttt{peanut\_oil} \rightsquigarrow \emptyset$ and $\Sigma_2 = \texttt{onions} \rightsquigarrow \texttt{leek}$ since $\sigma_2 = \texttt{leek} \rightsquigarrow \texttt{onions}$. According to the axiom $\texttt{green\_onion} \Rightarrow \texttt{onions}$ of O, the system further specializes the substitution Σ_2 into the substitution $\texttt{green\_onion} \rightsquigarrow \texttt{leek}$ and the user is proposed to replace green onions by leek in the recipe *Wonton Soup* and to suppress peanut oil. The generated adaptation path is $\texttt{AP} = \Sigma_1 \circ \Sigma_2$ (Fig. 1), with :

$$\begin{aligned}
\texttt{Sol}(\texttt{srce}) &= \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{green_onion} \wedge \ldots \wedge \texttt{peanut_oil} \wedge \textit{Nothing else} \\
\Sigma_2 = \texttt{green_onion} \rightsquigarrow \texttt{leek}, &\quad \Sigma_1 = \emptyset \rightsquigarrow \neg\,\texttt{peanut_oil} \\
\widetilde{\texttt{Sol}}(\texttt{tgt}) &= \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{leek} \wedge \ldots \wedge \neg\,\texttt{peanut_oil} \wedge \textit{Nothing else} \\
\texttt{tgt} &= \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{leek} \wedge \neg\,\texttt{peanut_oil}
\end{aligned}$$

The inferred solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ solves the initial query $\texttt{tgt} : \widetilde{\texttt{Sol}}(\texttt{tgt}) \models_O \texttt{tgt}$.

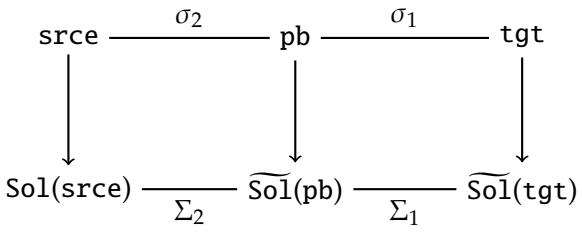


FIG. 1 – A similarity path and the associated adaptation path.

3.3 Why Learning Adaptation Knowledge in TAAABLE ?

In the version of the TAAABLE system that was proposed to participate in the CCC challenge, $\texttt{AKB} = \emptyset$ so adaptation knowledge is inferred from the ontology O. The main advantage of this approach lies in its simplicity : no external source of adaptation knowledge is needed and the system is able to propose a solution to any target problem. However, the system's adaptation capabilities (simple substitutions) appear to be very limited and the user has no means to give some feedback on the quality of the proposed adaptation.

For example, the substitution $\Sigma_1 = \emptyset \rightsquigarrow \neg$ `peanut_oil` suggests to remove the ingredient peanut oil in the retrieved recipe, but as the oil is used in this recipe to saute the bok choy, the adapted recipe turns out to be practically unfeasible. A better adaptation would suggest to replace peanut oil by e.g., sesame oil, which can be modeled by the substitution Σ_1 = `peanut_oil` $\rightsquigarrow$ `sesame_oil`. To generate this substitution automatically, the system could for example exploit the fact that the concepts `peanut_oil` and `sesame_oil` are both sub-concepts of the concept `oil` in $\mathcal{O}$. But still, some additional knowledge would be needed to express the fact that peanut oil should be replaced by sesame oil, and not by olive oil or hot chili oil, as `olive_oil` and `hot_chili_oil` are also sub-concepts of `oil` in $\mathcal{O}$. Besides, the system should be aware that this substitution is recommended *only in Asian cuisine*, which can be modeled by the more precise substitution Σ_1 = `asian` $\wedge$ `peanut_oil` $\rightsquigarrow$ `asian` $\wedge$ `sesame_oil`.

Furthermore, the second substitution Σ_2 = `green_onions` $\rightsquigarrow$ `leek` suggests to solely replace sliced green onions by uncooked leek. But the green onion was used in the original *Wonton Soup* for garniture, so the user might consider that raw leek added as garniture alters too much the taste of a soup. A better adaptation would consist in frying leek with e.g., tempeh and red bell pepper to prepare the garniture. Such an adaptation can be modeled by the substitution Σ_2 = `green_onions` $\rightsquigarrow$ `leek` $\wedge$ `tempeh` $\wedge$ `red_bell_pepper`. This substitution, which reflects a cooking know-how, can hardly be generated automatically from the ontology.

These examples show that in order to improve its adaptation capabilities, the system would greatly benefit from the availability of a set of adaptation rules that would capture more complex adaptation strategies. These adaptation rules cannot be generated automatically from the ontology and need to be acquired from other knowledge sources. These examples also show that the human expert plays a major role in adaptation knowledge acquisition and that in the cooking domain, adaptation rules are often highly contextual.

4 Opportunistic Adaptation Knowledge Discovery

The presented AK acquisition method combines two previous approaches of AK acquisition. The first one was implemented in the CabamakA system [7] and learns AK from differences between cases by the means of knowledge discovery techniques (section 4.1). The second one was implemented in the IakA system [5] and acquires adaptation knowledge at problem-solving time through interactions with the user (section 4.2).

4.1 Adaptation Knowledge Discovery from the Case Base

Machine learning algorithms aim at extracting some regularities from a set of observations. Knowledge discovery techniques combine efficient machine learning algorithms with human-machine interaction. In [7], AK is learned from differences between cases by the means of knowledge discovery techniques. A set of pairs of sources cases is taken as input of a frequent itemset extraction algorithm, which outputs a set of itemsets. Each of these itemsets can be interpreted as an adaptation rule. This approach of AK learning was motivated by the original idea proposed by Kathleen Hanney and Mark T. Keane in [10], in which the authors suggest that AK may be learned from differences between cases. The main assumption is that the differences that occur between cases in the case base are often representative of differences that will occur between future problems and the case base.

To learn adaptation rules from differences between cases, representing variations between cases is essential. In [3], expressive representation formalisms are proposed and it is shown that defining a partial order on the variation language can help organizing the learned rules by generality.

4.2 Opportunistic and Interactive Knowledge Acquisition

Experiential knowledge, or know-how, can often be acquired on-line, when users are using CBR tools. It is the aim of interactive and opportunistic knowledge acquisition strategies to support such

an acquisition. In these strategies, the system exploits its interactions with its user to build new pieces of knowledge, to test them and, in case of success, to retain them. Moreover, the knowledge acquisition process is often opportunistic, i.e, triggered by a previous reasoning failure : reasoning failures highlight missing knowledge and thus constitute a guidance for the acquisition process. A major advantage of interactive knowledge acquisition strategies is that they ensure that the user is in a favorable context when he participates to the acquisition process. In [4], a review of interactive and opportunistic knowledge acquisition approaches is proposed, and two strategies are developed. This work illustrates the efficiency of interactive and opportunistic knowledge acquisition approaches to acquire specific knowledge. On the other hand, it shows that such approaches only allow the systems to acquire small pieces of knowledge at a time.

4.3 Combining the two Approaches

When properly used, knowledge discovery techniques may have the strong advantage of automating a part of the knowledge acquisition process. In these approaches, dedicated human-machine interfaces allow the expert, through predefined interactions, to provide feedback on a set of suggestions generated automatically by the system. The role of the expert is thus reduced to the validation of a pre-selected set of knowledge pieces. The acquired knowledge is directly usable by the system, without the need for an additional formalization step. Automatic approaches also benefit from efficient machine learning algorithms that can be applied, as in [3], to learn adaptation rules at different levels of generality. However, these approaches still produce a large number of candidate knowledge units that have to be validated by a domain expert out of any context, which constitutes an important drawback.

Acquiring adaptation knowledge *offline*, i.e., independently of a particular problem-solving session, appears to be problematic. Offline AK acquisition forces the system's designer to anticipate the need for adaptation knowledge in problem-solving and to acquire it in advance, which can be very tedious, if not impossible. Offline acquisition of adaptation knowledge also makes difficult to come up with fine-grained adaptation rules, since adaptation knowledge is often highly contextual. For example, in the cooking domain, an egg can sometimes be substituted by 100 grams of tofu, but this adaptation rule may be applied only to certain types of dishes, like cakes or mayonnaise, and has proved to be irrelevant in order to adapt a mousse recipe or an omelet recipe. Acquiring such a rule would require to circumscribe its domain of validity in order to avoid over-generalization.

Moreover, initial acquisition of adaptation knowledge prevents the system from learning from experience. A CBR system with fixed adaptation knowledge has no way to improve its problem-solving capabilities, except by retaining in the case base a new experience each time a problem has been solved, as it is usually done in traditional CBR systems [8].

On the other hand, interactive and opportunistic knowledge acquisition approaches heavily rely on the human expert but ensure that the expert is "in context" when validating knowledge units that are to be acquired. Combining knowledge discovery techniques and interactive approaches, as it is proposed here, could overcome one of the limitations of KD by dramatically reducing the number of candidate adaptation rules presented to the expert. By triggering the process in an opportunistic manner, the expert is able to parametrize the KD in order to focus on specific knowledge to acquire in context. The resulting AK discovery process :

- is performed *on-line*, i.e., in the context of a problem-solving session,
- is *interactive* as adaptation knowledge is learned by the system through interactions with its user who acts as an expert,
- is *opportunistic* as it is triggered by reasoning failure, and, consequently, often helps repairing a failed adaptation,
- makes use of knowledge discovery techniques to provide *assistance* to the user in the formulation of new knowledge : the user is presented with a set of suggestions that are generated automatically from the case base.

5 Applying Opportunistic AK Discovery to TAAABLE

In this section, an opportunistic AK discovery process is applied to the context of the TAAABLE system.

5.1 AK Discovery

In TAAABLE, the AK discovery process consists in learning a set of substitutions from the case base by comparing two sets of recipes.

The Training Set. The training set TS is formed by selecting from the case base a set of pairs of recipes $(R_k, R_\ell) \in CB \times CB$ and by representing for each selected pair of recipes (R_k, R_ℓ) the variation $\Delta_{k\ell}$ from R_k to R_ℓ. The choice of the training set TS results from a set of interactions with the user during which he/she is asked to formulate the cause of the adaptation failure and to pick up a repair strategy.

Representing Variations. The variation $\Delta_{k\ell}$ from a recipe R_k to a recipe R_ℓ is represented in a language $\mathcal{L}_\Delta$ by a set of properties. Three properties a^-, a^+ and $a^=$ are defined in $\mathcal{L}_\Delta$ for each propositional variable a of $\mathcal{V}$, and $\Delta_{k\ell} \in \mathcal{L}_\Delta$ contains :

- the property a^- if $R_k \models_O a$ and $R_\ell \not\models_O a$,
- the property a^+ if $R_k \not\models_O a$ and $R_\ell \models_O a$,
- the property $a^=$ if $R_k \models_O a$ and $R_\ell \models_O a$.

For example, if :

$$R_k = \texttt{chinese} \wedge \texttt{soup} \wedge \ldots \wedge \texttt{peanut\_oil} \wedge \textit{Nothing else}$$
$$R_\ell = \texttt{chinese} \wedge \texttt{soup} \wedge \ldots \wedge \texttt{olive\_oil} \wedge \textit{Nothing else}$$

then $\Delta_{k\ell} = \{\texttt{chinese}^=, \texttt{soup}^=, \texttt{oil}^=, \texttt{peanut\_oil}^-, \texttt{olive\_oil}^+, \ldots\}$, provided that $\texttt{peanut\_oil} \models_O \texttt{oil}$, $\texttt{olive\_oil} \models_O \texttt{oil}$, $R_\ell \not\models_O \texttt{peanut\_oil}$ and $R_k \not\models_O \texttt{olive\_oil}$.

The inclusion relation $\subseteq$ constitutes a partial order on $\mathcal{L}_\Delta$ that can be used to organize variations by generality : a variation Δ is more general than a variation Δ' if $\Delta \subseteq \Delta'$.

Mining. The learning process consists in highlighting some variations $\Delta \in \mathcal{L}_\Delta$ that are more general than a "large" number of elements $\Delta_{k\ell}$ of TS. More formally, let

$$\texttt{support}(\Delta) = \frac{\texttt{card}\,\{\Delta_{k\ell} \in \texttt{TS} \mid \Delta \subseteq \Delta_{k\ell}\}}{\texttt{card TS}}$$

Learning adaptation rules aims at finding the $\Delta \in \mathcal{L}_\Delta$ such that $\texttt{support}(\Delta) \geq \sigma_s$, where $\sigma_s \in [0;1]$ is a learning parameter called the support threshold. It can be noticed that if $\Delta_1 \subseteq \Delta_2$ then $\texttt{support}(\Delta_1) \geq \texttt{support}(\Delta_2)$. The support threshold also has an influence on the number of generated variations. The number of generated variations increases when σ_s decreases. Thus, specifying a high threshold restricts the generation of variations to the most general ones, which can limit the number of generated variations and save computation time but has the effect to discard the most specific ones from the result set.

Each learned variation $\Delta = \{p_1, p_2, \ldots, p_n\} \in \mathcal{L}_\Delta$ is interpreted as a substitution of the form $A \rightsquigarrow B$ such that :

- $A \models_O a$ and $B \not\models_O a$ if $a^- \in \Delta$,
- $A \not\models_O a$ and $B \models_O a$ if $a^+ \in \Delta$,
- $A \models_O a$ and $B \models_O a$ if $a^= \in \Delta$.

For example, the variation $\Delta = \{\texttt{oil}^=, \texttt{peanut\_oil}^-, \texttt{olive\_oil}^+\}$ is interpreted as the substitution $\Sigma = \texttt{peanut\_oil} \rightsquigarrow \texttt{olive\_oil}$. The conjunct oil is not present neither in A nor in B since it is useless : $\texttt{peanut\_oil} \models_O \texttt{oil}$ and $\texttt{olive\_oil} \models_O \texttt{oil}$.

Filtering. For a retrieved recipe $\texttt{Sol(srce)}$, the result set can be filtered in order to retain only the substitutions $\Sigma = \texttt{A} \rightsquigarrow \texttt{B}$ that can be applied to modify $\texttt{Sol(srce)}$, i.e., such that $\texttt{Sol(srce)} \models_O \texttt{A}$.

Validation. Knowledge discovery aims at building a model of reality from a set of observations. But as a model of a part of reality is only valid with respect to a particular observer, any learned substitution has to be validated by a human expert in order to acquire the status of piece of knowledge.

5.2 Opportunistic Adaptation Knowledge Discovery

The AK discovery process turns the case base into an additional source of adaptation knowledge. This new source of knowledge is used during a problem-solving session to provide the CBR system with adaptation knowledge "on demand". A set of variations Δ is learned from the case base by comparing two sets of recipes and each learned variation Δ is interpreted as a substitution Σ that can be used to repair the adaptation path $\texttt{AP}$. Each learned substitution Σ is presented to the user for validation together with the corrected solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ resulting from its application. When the user validates the corrected solution, a new reformulation (σ, Σ) is added to the adaptation knowledge base $\texttt{AKB}$ so that the learned substitution Σ can be later reused to adapt new recipes. The AK discovery process is triggered either during the adaptation phase, to come up with suggestions of gradual solution refinements (see section 5.4 for an example), or during the solution test phase to repair a failed adaptation in response to the user's feedback (see section 5.5 for an example).

5.3 Implementation

To test the proposed adaptation knowledge acquisition method, a prototype was implemented that integrates the Taaable system [2] and the CabamakA system [7]. The case base contains 862 recipes taken from the CCC 2008 recipe set. The Taaable system is used to perform retrieval and adaptation. The CabamakA system is used to learn a set of substitutions Σ from the case base from the comparison of two sets of recipes. As in [7], the mining step is performed thanks to a frequent closed itemset extraction algorithm.

5.4 A First Example : Cooking a Chocolate Cake

An example is presented to illustrate how the case base is used as an additional source of adaptation knowledge. The AK discovery process is parametrized automatically and is used to provide assistance to the user by suggesting some gradual refinements for the proposed solution.

1. *Representing the Target Problem.* In this example, the user wants to cook a chocolate cake with baking chocolate and oranges. The target problem is :

$$\texttt{tgt} = \texttt{cake} \wedge \texttt{baking\_chocolate} \wedge \texttt{orange}$$

   In the Taaable interface, the field "Ingredients I Want" is filled in with the tokens $\texttt{baking\_chocolate}$ and $\texttt{orange}$ and the field "Types I Want" is filled in with the token $\texttt{cake}$.

2. *Retrieval.* The retrieval procedure generates the similarity path $\texttt{SP} = \sigma_1$ in which the substitution $\sigma_1 = \texttt{baking\_chocolate} \rightsquigarrow \texttt{chocolate}$ is generated automatically from the ontology $\mathcal{O}$ from the axiom $\texttt{baking\_chocolate} \Rightarrow \texttt{chocolate}$. $\texttt{SP}$ is applied to $\texttt{tgt}$ in order to produce the modified query $\texttt{srce} = \texttt{cake} \wedge \texttt{chocolate} \wedge \texttt{orange}$. The system retrieves the recipe *Ultralight Chocolate Cake*, whose representation $\texttt{Sol(srce)}$ is :

$$\texttt{Sol(srce)} = \texttt{cake} \wedge \texttt{cocoa} \wedge \texttt{orange} \wedge \ldots \wedge \textit{Nothing else}$$

   Since the ontology $\mathcal{O}$ contains the axiom $\texttt{cocoa} \Rightarrow \texttt{chocolate}$, $\texttt{Sol(srce)}$ solves the query $\texttt{srce}$: $\texttt{Sol(srce)}$ is such that $\texttt{Sol(srce)} \models_O \texttt{srce}$.

3. *Adaptation.* AKB is assumed to be empty, so to construct the adaptation path AP, the substitution chocolate $\rightsquigarrow$ baking_chocolate is generated automatically from σ_1. This substitution is further specialized into the substitution Σ_1 = cocoa $\rightsquigarrow$ baking_chocolate, according to the axiom cocoa $\Rightarrow$ chocolate of $\mathcal{O}$. A first solution $\widetilde{\text{Sol}}$(tgt) is computed by applying to Sol(srce) the adaptation path AP = Σ_1. The user suggests that an ingredient is missing in $\widetilde{\text{Sol}}$(tgt) but could not identify a repair strategy. An AK discovery is triggered in order to suggest gradual refinements of $\widetilde{\text{Sol}}$(tgt).
4. *Choosing the Training Set.* The training set TS is chosen from Σ_1 : AK is learned by comparing the recipes containing cocoa with the recipes containing baking chocolate. TS is composed of the set of variations $\Delta_{k\ell} \in \mathcal{L}_\Delta$ between pairs of recipes $(R_k, R_\ell) \in CB \times CB$ such that $\{\text{cocoa}^-, \text{baking\_chocolate}^+\} \subseteq \Delta_{k\ell}$.
5. *Mining and Filtering.* A value is given to the support threshold σ_s and the mining step outputs a set of variations. A filter retains only the variations that correspond to substitutions applicable to modify Sol(srce).
6. *Solution Test and Validation.* The user selects the learned variation $\Delta = \{\text{cocoa}^-, \text{baking\_chocolate}^+, \text{oil}^-\}$ from the result set. Δ is interpreted as the substitution Σ = cocoa $\wedge$ oil $\rightsquigarrow$ baking_chocolate, which suggests to replace cocoa by baking chocolate in the retrieved recipe and to remove oil. The user explains this rule by the fact that baking chocolate contains more fat than cocoa, and therefore substituting cocoa by baking chocolate implies to reduce the quantity of fat in the recipe.

   Further solution refinements are proposed to the user. The set of learned variations is filtered in order to retain only the substitutions Δ' that are more specific than Δ, i.e., such that $\Delta \subseteq \Delta'$. Among the retained variations is the variation $\Delta' = \{\text{cocoa}^-, \text{baking\_chocolate}^+, \text{oil}^-, \text{vanilla}^-\}$, which is interpreted as the substitution Σ' = cocoa $\wedge$ oil $\wedge$ vanilla $\rightsquigarrow$ baking_chocolate. Σ' suggests to also remove vanilla in the recipe *Ultralight Chocolate Cake*.

   The user is satisfied with the refined solution $\widetilde{\text{Sol}}$(tgt) resulting from the application of the adaptation path AP = Σ' to Sol(srce), so the reformulation (baking_chocolate $\rightsquigarrow$ chocolate, cocoa $\wedge$ oil $\wedge$ vanilla $\rightsquigarrow$ baking_chocolate) is added to the adaptation knowledge base AKB.

5.5 A Second Example : Cooking a Chinese Soup

A second example is presented in which the AK discovery process is triggered in response to the user feedback in order to repair the adaptation presented in Sect. 3. In this example, the user is encouraged to formulate the cause of the adaptation failure. A repair strategy is chosen that is used to parametrize the AK discovery process.

1. *Representing the Target Problem.* In this example, the target problem tgt is :

   $$\text{tgt} = \text{chinese} \wedge \text{soup} \wedge \text{leek} \wedge \neg\, \text{peanut\_oil}$$

   In the Taaable interface, the field "Ingredients I Want" is filled in with the token leek, the field "Ingredients I Don't Want" is filled in with the token peanut_oil and the field "Types I Want" is filled in with the tokens chinese and soup.
2. *Retrieval.* As in Sect. 3, two substitutions $\sigma_1 = \neg\, \text{peanut\_oil} \rightsquigarrow \emptyset$ and σ_2 = leek $\rightsquigarrow$ onions are generated automatically from the ontology $\mathcal{O}$. The similarity path SP = $\sigma_2 \circ \sigma_1$ is applied to tgt in order to produce the modified query srce = chinese $\wedge$ soup $\wedge$ onions. The system retrieves the recipe *Wonton Soup*, whose representation Sol(srce) solves the query srce : Sol(srce) is such that Sol(srce) $\models_{\mathcal{O}}$ srce.
3. *Adaptation.* Initially, AKB = $\emptyset$, so to construct the adaptation path AP, two substitutions $\Sigma_1 = \emptyset \rightsquigarrow \neg\, \text{peanut\_oil}$ and Σ_2 = green_onion $\rightsquigarrow$ leek are automatically generated from σ_1 and σ_2.

4. *Solution Test and Validation.* The solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ is presented to the user for validation, together with the adaptation path $\texttt{AP} = \Sigma_1 \circ \Sigma_2$ that was used to generate it.
5. *The User is Unsatisfied !* The user complains that the adapted recipe is practically unfeasible because the proposed solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ does not contain oil anymore, and oil is needed to saute the bok choy.
6. *What has Caused the Adaptation Failure ?* The cause of the adaptation failure is identified through interactions with the user. The user validates the intermediate solution $\widetilde{\texttt{Sol}}(\texttt{pb})$ that results from the application of the substitution $\Sigma_2 = \texttt{green\_onion} \rightsquigarrow \texttt{leek}$ to $\texttt{Sol}(\texttt{srce})$. But the user invalidates the solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ that results from the application of $\Sigma_1 = \emptyset \rightsquigarrow \neg\,\texttt{peanut\_oil}$ to $\widetilde{\texttt{Sol}}(\texttt{pb})$. The substitution Σ_1 is identified as responsible for the adaptation failure since its application results in the removal of oil in the recipe.
7. *Choosing a Repair Strategy.* A repair strategy is chosen according to the user's feedback. The user expresses the need for oil in the adapted recipe, so the repair strategy consists in replacing peanut oil by another oil. An AK discovery process is triggered to decide which oil to replace peanut oil with.
8. *Choosing the Training Set.* A set of recipes that contain peanut oil is compared with a set of recipes containing other types of oil. The training set TS is composed of the set of variations $\Delta_{k\ell} \in \mathcal{L}_\Delta$ between pairs of recipes $(\texttt{R}_k, \texttt{R}_\ell) \in \texttt{CB} \times \texttt{CB}$ such that $\{\texttt{oil}^=, \texttt{peanut\_oil}^-\} \subseteq \Delta_{k\ell}$.
9. *Mining and Filtering.* A value is given to the support threshold σ_s and the mining step outputs a set of variations. A filter retains only the variations that correspond to substitutions applicable to modify $\texttt{Sol}(\texttt{pb})$.
10. *Solution Test and Validation.* The user selects the learned variation $\Delta = \{\texttt{oil}^=, \texttt{peanut\_oil}^-, \texttt{olive\_oil}^+\}$ from the result set. Δ is interpreted as the substitution $\Sigma = \texttt{peanut\_oil} \rightsquigarrow \texttt{olive\_oil}$, which suggests to replace peanut oil by olive oil in the retrieved recipe. The adaptation path $\texttt{AP} = \Sigma \circ \Sigma_2$ is computed and the repaired solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ is presented to the user for validation. The user is satisfied with the corrected solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$, so the reformulation ($\emptyset \rightsquigarrow \neg\,\texttt{peanut\_oil}$, $\texttt{peanut\_oil} \rightsquigarrow \texttt{olive\_oil}$) is added to the adaptation knowledge base AKB.

6 Discussion and Related Work

AK acquisition is a difficult task that is recognized to be a major bottleneck for CBR system designers due to the high knowledge-engineering costs it generates. To overcome these knowledge-engineering costs, a few approaches (e.g., [7, 6, 10]) have applied machine learning techniques to learn AK offline from differences between cases of the case base. In [10], a set of pairs of source cases is selected from the case base and each selected pair of source cases is considered as a specific adaptation rule. The featural differences between problems constitute the antecedent part of the rule and the featural differences between solutions constitute the consequent part. Michalski's closing interval rule algorithm is then applied to generalize adaptation rule antecedents. In [6], adaptation knowledge takes the form of a set of adaptation cases. Each adaptation case associates an adaptation action to a representation of the differences between the two source problems. Machine learning algorithms like C4.5 or RISE are applied to learn generalized adaptation knowledge from these adaptation cases in order to improve the system's case-based adaptation procedure.

When applying machine learning techniques to learn adaptation knowledge from differences between cases, one main challenge concerns the choice of the training set : which cases are worth comparing ? Arguing that (1) the size of the training set should be reduced to minimize the cost of the adaptation rule generation process and that (2) the source cases that are worth comparing should be the ones that are more similar, only the pairs of source cases that were judged to be similar according to a given similarity measure are selected in [6] and [10]. However, committing to a particular similarity

measure might be somewhat arbitrary. Therefore, in [7], the authors decided to include in the training set all the pairs of distinct source cases of the case base. This paper introduces a third approach : the choice of the training set is determined interactively and according to the problem-solving context, taking advantage of the fact that the AK discovery process is triggered on-line. This approach appears to be very promising since the learning algorithm can be parametrized in order to learn only the knowledge that is needed to solve the target problem.

The examples presented above also show that knowledge discovery techniques allow to come up with more complex adaptation strategies than the simple one-to-one ingredient substitutions generated from the ontology O. In particular, these techniques can help identifying interactions between the different ingredients that appear in the recipes (like e.g., that cocoa contains less fat than baking chocolate, so oil should be removed) as well as co-occurrences of ingredients (like say, that cinnamon is well-suited with apples). Besides, adaptation knowledge is learned at different levels of generality, so the user can be guided into gradual solution refinements.

Several CBR systems make use of interactive and/or opportunistic knowledge acquisition approaches to improve their learning capabilities. For example, in Creek, an approach that combines case-based and model-based methods, general knowledge is acquired through interactions with the user [1]. This knowledge acquisition process is provided in addition to the traditional case acquisition and allows the system to acquire knowledge that cannot be captured through cases only. In the Dial system, adaptation knowledge is acquired in the form of adaptation cases : when a case has to be adapted, the adaptation process is memorized in the form of a case and can be reused to adapt another case. Hence, adaptation knowledge is acquired through a CBR process inside the main CBR cycle. It must be remarked that adaptation cases can either be built automatically by adaptation of previous adaptation cases or manually by a user who interactively builds the adaptation case in response to a problem by selecting the appropriates operations to perform [11]. Hence, knowledge acquisition in Dial appears to be both interactive and opportunistic. Chef is obviously related to the work described here [9]. Chef is a case-based planner in the cooking domain, its task is to build recipes on the basis of a user's request. The input of the system is a set of goals (tastes, textures, ingredients, types of dishes) and the output is a plan for a single recipe that satisfies all the goals. To solve this task, Chef is able to build new plans from old ones stored in memory. The system is provided with the ability to choose plans on the basis of the problems that they solve as well as the goals they satisfy, but it is also able to predict problems and to modify plans to avoid failures (plans are indexed in memory by the problems they avoid). Hence, Chef learns by providing causal explanations of failures thus marking elements as "predictive" of failures. In other words, the acquired knowledge allows the system to avoid identical failures to occur again. In our approach, we propose to go one step further by using failure to acquire knowledge that can be more widely used.

7 Conclusion and Future Work

In this paper, a novel approach for adaptation knowledge acquisition is presented in which the knowledge learned at problem-solving time by knowledge discovery techniques is directly reused for problem-solving. An application is proposed in the context of the cooking CBR system Taaable and the feasibility of the approach is demonstrated on some use cases. Future work will include developing a graphical user interface and doing more extensive testing. Opportunistic and interactive knowledge discovery in Taaable implies that the user plays the role of the domain expert, which raises several issues. For example, how to be sure that the knowledge expressed by a particular user is valuable ? How to ensure that the adaptation knowledge base will remain consistent with time ? Besides, Taaable is meant to be multi-user, so if the system's knowledge evolves with experience, some synchronization problems might occur. Therefore, the envisioned multi-user, ever-learning Taaable system needs to be thought of as a collaborative tool in which knowledge acquired by some users can be revised by others.

Aknowledgements

The authors wish to thank the reviewers of the paper for their interesting remarks and suggestions. Some of them could not be taken into account in this paper since they point out long-term issues that would require more research.

Références

[1] A. Aamodt. Knowledge-Intensive Case-Based Reasoning in CREEK. In P. Funk et P. A. Gonzàlez-Calero, editor, *Proceedings of the European Conference on Case-Based Reasoning, ECCBR'04*, volume 3155 of *Lecture Notes in Artificial Intelligence*, pages 1–15. Springer, 2004.

[2] F. Badra, R. Bendaoud, R. Bentebibel, P.-A. Champin, J. Cojan, A. Cordier, S. Després, S. Jean Daubias, J. Lieber, T. Meilender, A. Mille, E. Nauer, A. Napoli, et Y. Toussaint. TAAABLE : Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In Martin Schaaf, editor, *9th European Conference on Case-Based Reasoning - ECCBR 2008*, pages 219–228, Trier Allemagne, 2008.

[3] F. Badra et J. Lieber. Representing Case Variations for Learning General and Specific Adaptation Rules. In Amedeo Cesta et Nikos Fakotakis, editors, *Fourth Starting AI Researcher's Symposium (STAIRS 2008)*, pages 1–11, Patras Grèce, 2008. IOS Press.

[4] A. Cordier. *Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning*. PhD thesis, Université Lyon 1, 2008.

[5] A. Cordier, B. Fuchs, L. Lana de Carvalho, J. Lieber, et A. Mille. Opportunistic acquisition of adaptation knowledge and cases - the IakA approach. In *Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008, Trier, Germany, September 1-4, 2008. Proceedings*, pages 150–164, 2008.

[6] S. Craw, N. Wiratunga, et R. Rowe. Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170(16-17) :1175–1192, 2006.

[7] M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, et L. Szathmary. Case base mining for adaptation knowledge acquisition. In *Proceedings of the International Conference on Artificial Intelligence, IJCAI'07*, pages 750–756, 2007.

[8] R. L.ópez de Mántaras et E. Plaza. Case-based reasoning : An overview. *AI Communications*, 10(1) :21–29, 1997.

[9] K. J. Hammond. CHEF : A model of case-based planning. In *AAAI*, pages 267–271, 1986.

[10] K. Hanney et M. T. Keane. The adaption knowledge bottleneck : How to ease it by learning from cases. In *Case-Based Reasoning Research and Development, Second International Conference, (ICCBR'97)*, pages 359–370, 1997.

[11] D. B. Leake, A. Kinley, et D.C. Wilson. Acquiring Case Adaptation Knowledge : A Hybrid Approach. In *AAAI/IAAI*, volume 1, pages 684–689, 1996.

[12] J. Lieber. Le moteur de raisonnement à partir de cas de WikiTaaable. In *À paraître dans les actes du 17ème atelier raisonnement à partir de cas (RàPC-09)*, 2009.

[13] E. Melis, J. Lieber, et A. Napoli. Reformulation in case-based reasoning. In B. Smyth et P. Cunningham, editor, *Proc. of the Fourth European Workshop on Case-Based Reasoning, (EWCBR'98)*, Lecture Notes in Artificial Intelligence 1488, pages 172–183. Springer, 1998.

[14] M. Schaaf, editor. *ECCBR 2008, The 9th European Conference on Case-Based Reasoning, Trier, Germany, September 1-4, 2008, Workshop Proceedings*, 2008.

WIKITAAABLE, un wiki sémantique utilisé comme un tableau noir dans un système de raisonnement à partir de cas textuel

Amélie Cordier[1], Jean Lieber[2], Pascal Molli[2],
Emmanuel Nauer[2], Hala Skaf-Molli[2], Yannick Toussaint[2]

[1]Université de Lyon, CNRS,
Université Lyon 1, LIRIS, UMR5205, F-69622, France.
{firstname.lastname}@liris.cnrs.fr

[2]INRIA Nancy-Grand Est
Nancy Université
LORIA – UMR 7503. B.P. 239, F-54506 Vandœuvre-lès-Nancy cedex, France.
{firstname.lastname}@loria.fr

Résumé

Les wikis sémantiques permettent à des communautés d'utilisateurs de produire des connaissances formalisées compréhensibles et utilisables par les machines. Afin d'aller plus loin, on pourrait imaginer d'utiliser les wikis sémantiques comme des tableaux noirs permettant aux humains et aux machines d'interagir afin de construire des connaissances qui soient à la fois utiles aux humains et exploitables par les machines. Dans ce papier, nous présentons une étude de cas portant sur l'utilisation d'un wiki sémantique (connu sous le nom de Semantic Media Wiki) comme un tableau noir pour gérer des données et des connaissances culinaires. Cette étude de cas est réalisée dans le contexte de TAAABLE, le fameux système de raisonnement à partir de cas en ligne capable de résoudre des problèmes culinaires en utilisant une base de recettes existantes. Avec WIKITAAABLE, l'évolution de TAAABLE intégrant un wiki sémantique, nous voulons montrer comment un wiki sémantique peut apporter support et assistance aux utilisateurs chargés des tâches fastidieuses de gestion des connaissances. Nous montrons en particulier comment le wiki facilite la prise en compte du retour d'expérience (*feedback*) des utilisateurs. Les questions relatives à la présence d'utilisateurs multiples ainsi qu'à l'intégration de plusieurs mécanismes de gestion des connaissances dans une application unique sont discutées à la fin de ce papier.

Mot-clés : Acquisition collaborative de connaissances, wikis sémantiques, annotation et indexation textuelle, ontologies, raisonnement à partir de cas (RÀPC), coopération homme-machine, enrichissement d'ontologies.

1 Introduction

Par le passé, les wikis ont démontré leurs capacités à transformer une communauté d'étrangers en communauté de collaborateurs. En intégrant des technologies issues du Web Sémantique, les wikis sémantiques [8, 7, 2] permettent désormais à ces communautés de contributeurs de produire des connaissances formalisées exploitables par les machines. Il est fort probable que la prochaine étape soit d'utiliser les wikis sémantiques comme des tableaux noirs permettant aux humains et aux machines d'interagir plus facilement pour produire des connaissances partagées, compréhensibles et utilisables par tous, humains et machines.

Forts de cette observation, dans ce papier nous étudions la question en la développant dans le cadre du projet TAAABLE [1]. TAAABLE[1] permet à ses utilisateurs d'envoyer des requêtes culinaires

[1]TAAABLE est accessible à l'adresse `http://taaable.fr/`

auxquelles le système répondra en s'appuyant sur une base de recettes de cuisine. Par exemple, on peut demander à TAAABLE une recette de dessert à la rhubarbe mais qui ne contienne pas de chocolat. Si aucune recette de la base de recette ne permet de répondre à cette requête, TAAABLE en choisit une proche (par exemple, la recette de la tarte rhubarbe-chocolat) et l'adapte afin de répondre à la requête posée (par exemple, en retirant le chocolat, ou encore en le remplaçant par un autre ingrédient, comme les myrtilles ou les framboises qui se marient à merveille avec la rhubarbe et apportent l'équilibre en sucre nécessaire [2]). Comme tout lecteur attentif s'en doute, le moteur de raisonnement de TAAABLE s'appuie sur les principes du raisonnement à partir de cas (RÀPC [6]) et pratique donc couramment l'adaptation. Le moteur de TAAABLE utilise différentes sources de données et de connaissances : un ensemble de recettes annotées (la base de recettes) ; des ontologies pour les ingrédients, types de plats, origines géographiques des plats, types de régimes alimentaires (végétarien, sans oléagineux, sans alcool) ; et enfin, un ensemble de règles d'adaptation. Les recettes annotées sont élaborées à partir des recettes initiales, (souvent mal) écrites en langue naturelle : un outil d'annotation utilise les différentes ontologies pour produire ces recettes annotées.

Lorsque le système fonctionne bien (autant dire jamais), la maintenance de la base de recettes et des connaissances périphériques est assez facile. En revanche, s'il est nécessaire de faire des corrections ou des mises à jour un peu compliquées, les problèmes commencent à apparaître. Par exemple, si l'on souhaite ajouter une recette à la base, il faut que cette recette puisse être indexée comme les autres sans quoi elle ne sera jamais utilisée. Par ailleurs, si l'on détecte une erreur dans les résultats produits par le système, il serait souhaitable de pouvoir la corriger immédiatement plutôt que de la noter sur un petit cahier en attendant de la corriger dans la version de l'année suivante. Ces différents points sont des limites évidentes de la première version de TAAABLE : le processus d'indexation des recettes en langage naturel est loin d'être infaillible, le système n'offre aucune possibilité d'interactions en cas d'erreur et, pour couronner le tout, même si toutes les requêtes sont tracées, aucune prise en compte du retour d'expériences (*feedback*) des utilisateurs n'est effectuée (expression de la satisfaction, commentaires, etc.). En résumé, à l'exception des corrections apportées par les concepteurs, qui elles-mêmes ne sont pas aisées, rien ne permet au système TAAABLE d'améliorer ses capacités d'adaptation internes.

Dans ce papier, nous décrivons l'utilisation d'un wiki sémantique dans TAAABLE et nous étudions sa capacité à nous aider à gérer les données (les recettes et les informations terminologiques du domaine culinaire) et les connaissances (les ontologies et les règles d'adaptation) utilisées par le moteur de RÀPC. La vocation finale du wiki sémantique est d'aider les utilisateurs à produire de nouvelles connaissances : ajout de nouvelles recettes, correction des problèmes d'indexation existants, prise en compte du *feedback* sur les résultats de l'adaptation. Le wiki rassemble également les processus permettant d'effectuer des adaptations, d'indexer de nouvelles recettes et de prendre en compte le *feedback* des utilisateurs pour améliorer les capacités internes d'adaptation et d'indexation. Même si l'objectif final est de permettre à tous les membres d'une communauté de collaborer et de contribuer à la construction des connaissances de TAAABLE, le wiki est pour l'instant utilisé uniquement par les concepteurs de l'application et apporte déjà une aide significative.

Ce papier est découpé en deux sections principales, l'une portant sur la « vieille » application TAAABLE (section 2) et l'autre sur la nouvelle application WIKITAAABLE (section 3), l'idée étant de montrer en quoi la nouvelles version apporte des solutions aux problèmes soulevés par l'ancienne. L'article se termine par une conclusion ainsi qu'une rapide discussion sur les travaux en cours et à venir.

2 Le projet TAAABLE

Le projet TAAABLE a participé en 2008 au Computer Cooking Contest[3] (CCC), une compétition scientifique internationale visant à faire s'affronter des systèmes informatiques capables de résoudre des problèmes de cuisine. Selon les règles du concours, les systèmes candidats devaient être à même de

[2] Recette disponible sur demande. Nous vous la recommandons, c'est fameux

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

répondre à des requêtes (du type de celle présentée en introduction relative à la tarte à la rhubarbe) en retrouvant et/ou en adaptant des recettes existantes afin qu'elles satisfassent les contraintes imposées par les utilisateurs. Le livre de recettes, identique pour tous les participants, était constitué d'un ensemble de recettes textuelles décrites simplement par un titre, une liste faiblement structurée d'ingrédients et un ensemble d'instructions de préparation. Les requêtes des utilisateurs, quant à elles, peuvent porter non seulement sur les ingrédients à inclure ou à éviter dans la recette mais peuvent aussi poser des contraintes sur les types de plats, leurs origines ou encore les régimes alimentaires à prendre en compte. Par exemple : « je veux une salade méditerranéenne végétarienne, sans produits à base de noix, contenant des tomates mais pas d'ail et encore moins de concombres » est une requête qui aurait pu figurer parmi les exercices imposés aux participants du CCC en 2008.

Pour résoudre de tels problèmes, l'assemblage de différentes compétences est souhaitable. C'est pourquoi le projet TAAABLE implique des chercheurs qui s'intéressent à des champs de recherches diversifiés tels que le raisonnement à partir de cas (RÀPC), l'ingénierie des ontologies, la fouille de données et de textes, l'indexation de ressources textuelles ou encore la classification hiérarchique. Grâce à la fusion de ces différentes compétences, l'équipe est parvenue à réaliser un système fonctionnel qui a obtenu le titre de vice-champion du CCC en 2008. Pour 2009, les règles du nouveau CCC n'ont que très sensiblement été modifiées, nous avons donc décidé de réutiliser notre expérience pour cette nouvelle édition. Par conséquent, nos objectifs et nos recherches en cours visent désormais à améliorer le système existant en développant de nouvelles possibilités d'évolution de l'application.

Passons aux choses sérieuses. Le reste de cette section présente une vision générale de l'architecture de TAAABLE (ancienne version) et décrit ses principales composantes en en montrant les limites.

2.1 L'application TAAABLE : une vision générale

L'architecture du système TAAABLE est composée de deux parties distinctes. La partie dite *offline* couvre les différents aspects de la gestion des connaissances et l'annotation des recettes du CCC. Des outils spécifiques ont été développés pour les besoins de cette partie. Cette partie se présente donc comme une chaîne de traitement à l'issue de laquelle est obtenu un ensemble de données et de connaissances utilisées par le moteur de raisonnement à partir de cas de l'application.

Ce moteur, associé à l'interface web d'interrogation et de présentation des résultats constitue la partie dite *online* de l'application. La figure 1 reprend ces différents éléments.

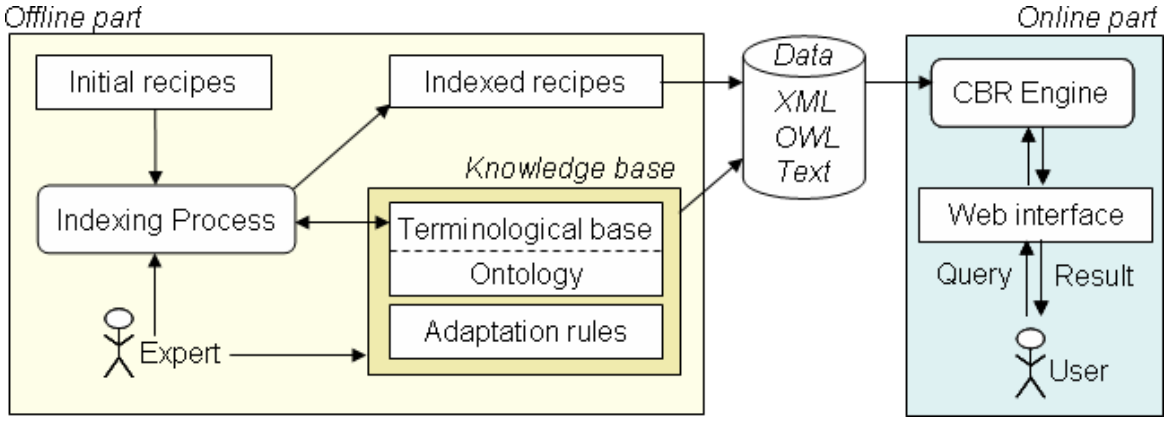


FIG. 1 – Architecture du système TAAABLE.

L'interface web de l'application permet aux utilisateurs de saisir leurs requêtes. Le système répond à une requête en retournant une liste de recettes jugées satisfaisantes. Si des adaptations de ces recettes sont nécessaires, elles sont signifiées à l'utilisateur qui peut alors en consulter le détail. La figure 2 montre un exemple de réponse à la requête « je vous saurai gré de bien vouloir me proposer une recette de tarte à l'orange ».

FIG. 2 – L'interface web de l'application de RÀPC TAAABLE

Pour TAAABLE, adapter une recette consiste à remplacer certains ingrédients par d'autres. En cliquant sur le titre d'une recette, l'utilisateur peut en voir le contenu détaillé, la liste des ingrédients initiaux et surtout les ingrédients à substituer par d'autres. Sur la figure 3, la recette intitulée *Key Lime Pie* est adaptée en remplaçant le *key lime juice* et/ou le *key lime* par de l'orange (vision simpliste mais efficace de l'adaptation).

2.2 Organisation des connaissances

Hiérarchie des ingrédients. Le moteur de raisonnement à partir de cas adapte les recettes en substituant des ingrédients par d'autres. Pour l'instant, pour des raisons de simplicités, chaque ingrédient est remplacé par au plus un autre ingrédient. L'idée sous-jacente à cette démarche est qu'en général, une recette peut être adaptée en remplaçant un ingrédient par un autre ingrédient en étant « proche » de celui-ci. Toute la question est donc de savoir ce qu'est un ingrédient « proche ». Pour cela, TAAABLE utilise une ontologie qui organise les ingrédients en fonction des besoins de l'application (et non en fonction des propriétés botaniques des légumes par exemple...). Le moteur de raisonnement à partir de cas est capable de calculer des « coûts de substitution » en fonction de cette ontologie : plus les ingrédients sont proches dans l'ontologie et plus le coût de substitution de l'un par l'autre est faible. Par exemple, le coût de remplacement d'un orange par un citron est plus faible que le coût de remplacement d'une orange par un steak haché car, selon l'ontologie utilisée, l'orange est plus proche

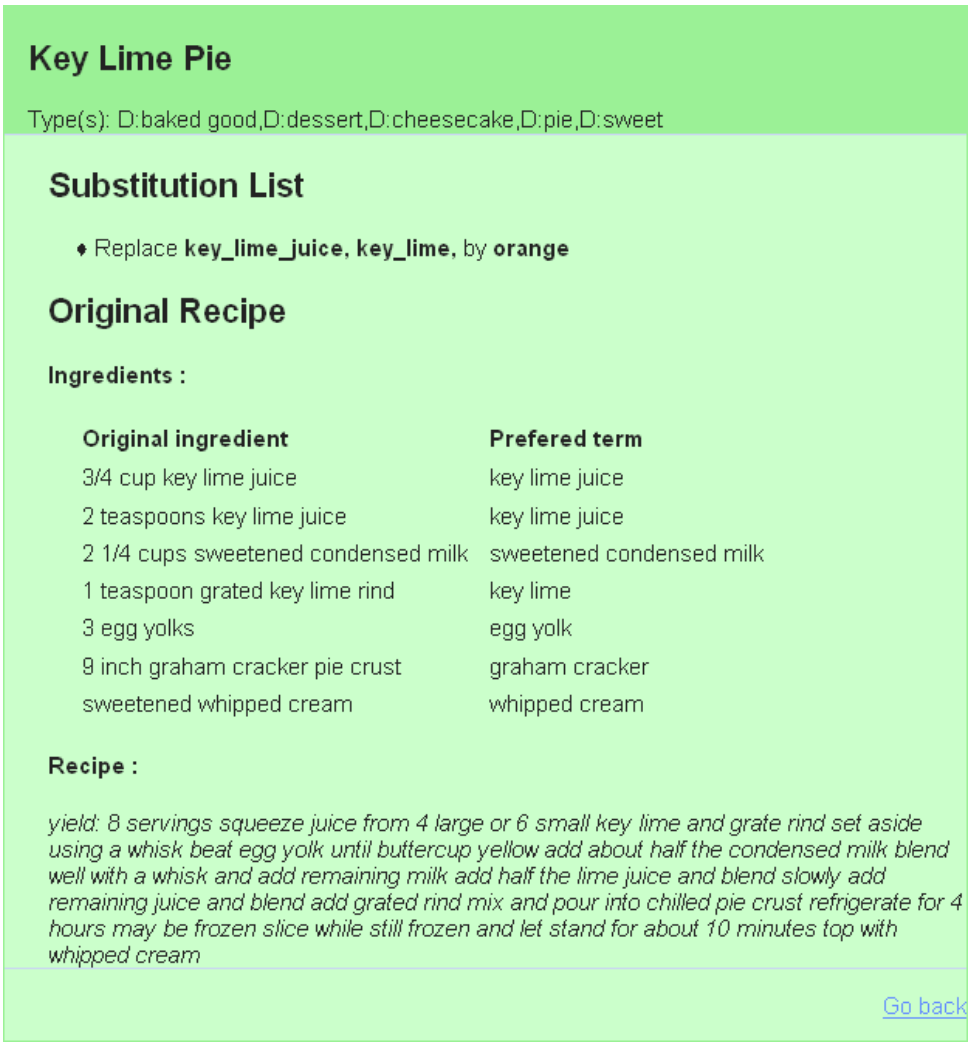


FIG. 3 – Un exemple d'adaptation d'une recette.

du citron que du steak haché.

Hiérarchies des types de recettes. Le typage des recettes est nécessaire afin de permettre de répondre aux requêtes du CCC. En effet, les origines des recettes, par exemple, ne sont pas précisées dans les recettes initiales et doivent être ajoutées à la main ou bien inférées d'une façon ou d'une autre. Afin de typer les recettes, trois hiérarchies complémentaires ont été définies. La hiérarchie des *dish moments* (les moments du plat) tels que les gâteaux apéritifs ou encore les desserts, la hiérarchie des *dish types* (les types de plats) comme les gâteaux ou les pizzas, et la hiérarchie des *dish origins* (les origines des plats) qui contient des concepts tels que « Méditerranéen » ou « Chinois ».

Dans la suite, nous parlons des quatre hiérarchies que nous venons de décrire comme de *l'ontologie* du système. L'ontologie de TAAABLE décrit donc les concepts du domaine utilisés dans l'application. Le lien entre la structure conceptuelle (l'ontologie) et le niveau linguistique (les recettes originales) est établi grâce à la base terminologique : chaque concept de l'ontologie est associé à sa forme linguistique qui se présente comme un ensemble de variantes terminologiques. Par exemple, le concept *litchi* est associé aux termes *litchi*, *lichi*, *lychee*, *leechee*, et *laichee*.

L'ontologie et la base terminologique ont été construites conjointement par les experts. En fonction des résultats du processus d'indexation (cf. ci-dessous pour plus de détails) qui peut mettre en évidence des problèmes dans la terminologie, les experts pouvaient décider « au cas par cas » quels termes et quels concepts (éventuellement associés à des termes) devaient être ajoutés dans la hiérarchie.

En complément de l'ontologie, le système de RÀPC peut être amené à utiliser une base de règles d'adaptation pour adapter les recettes. Une règle d'adaptation est définie comme une paire ordonnée d'ensemble d'ingrédients (s_1, s_2) associée à un coût établi par l'expert. Formellement, (s_1, s_2) signifie que l'ensemble d'ingrédients s_1 peut être remplacé par l'ensemble d'ingrédients s_2.

Les règles d'adaptation et l'ontologie constituent la **base de connaissances** du système.

2.3 Annoter les recettes en fonction de la base terminologique

Le **processus d'annotation** est semi-automatique et s'appuie sur les données disponibles dans la base terminologique. En entrée, il prend les recettes dans leur forme textuelle initiale. Pour chaque recette, la sortie du processus contient : (1) une version annotée de la recette au format XML initial mais avec des balises additionnelles, (2) une liste des concepts permettant d'indexer la recette exprimés en logique propositionnelle et (3) un rapport d'erreur mettant en évidence les lignes d'ingrédients de la recette initiale qui n'ont pas pu être annotées en utilisant la base terminologique (ce rapport est très informatif sur les failles éventuelles de la base utilisée). Ces trois points sont détaillés ci-après.

2.3.1 Recettes annotées.

Comme nous l'avons vu plus haut, les recettes du CCC sont représentées en XML mais sont très faiblement structurées. Des balises sont utilisées pour identifier le titre de la recette (élément `TI`), les ingrédients (éléments `IN`) et le corps de texte décrivant la préparation (élément `PR`). Le processus d'annotation permet donc d'ajouter des balises précisant chaque ingrédient de la recette. Il recherche dans chaque ligne d'ingrédient la présence de termes de la base terminologique et introduit, pour chaque terme trouvé, le concept impliqué. Par exemple, la ligne d'ingrédient `<IN>300 g mashed bananas</IN>` sera annoté comme : `<IN><ING>banana</ING><QT>300</QT><U>g</U><QL>mashed</QL><R/> </IN>` où `<ING>banana</ING>` est le concept de l'ontologie associé au terme *bananas*, `<QT>300</QT>` est la quantité, `<U>g</U>` est l'unité, `<QL>mashed</QL>` est un "qualifier" et `<R/>` est le reste de la ligne d'ingrédient, c'est-à-dire l'ensemble des éléments qui n'ont pas été reconnus par l'analyseur étant données les informations contenues dans la base terminologique. Dans l'exemple présent, comme tout a été reconnu, le reste est vide. La ligne ainsi annotée est utilisée par les experts pour contrôler la correction de l'étape d'analyse.

Les différents types (types de plats, origines, moments des plats) d'une recette n'apparaissent pas explicitement dans la description initiale de la recette. Ces types doivent être inférés à partir des éléments disponibles. Dans TAAABLE, ceci est réalisé par un processus en trois étapes. Tout d'abord, on recherche si une recette ayant le même titre existe sur le site *recipesource.com*. Si tel est le cas, on récupère le maximum d'informations de typage depuis ce site. Si cette première étape échoue, on recherche si le titre de la recette contient des éléments qui en eux-mêmes représentent un type de plat ou une origine (par exemple, le *Banana Butterfinger cake* est sans aucun doute un *cake*). Enfin, si l'étape 2 échoue également, on utilise un ensemble de règles d'association du type *ingredient(s)* $\rightarrow$ *type ou origine* (par exemple, *mascarpone* $\wedge$ *coffee* $\rightarrow$ *tiramisu*) afin de typer la recette. Les règles d'association proviennent en partie d'une extraction réalisée sur l'ensemble des ressources disponibles dans la base de données du site *recipesource.com*, les autres règles sont définies manuellement en fonction des besoins observés.

L'annotation des recettes par les types est un processus relativement bruité. 30% des recettes n'ont aucun type affecté, certains types sont manquants (en particulier les moments des plats) et il y a certaines erreurs (par exemple, *pizza sauce* n'est pas vraiment une *pizza*). Afin d'éviter les comportements anormaux du système, les experts doivent vérifier manuellement les annotations de chaque recette.

2.3.2 Indexation des recettes sous forme propositionnelle.

Le langage de représentation des connaissances utilisé par le moteur de RÀPC est un fragment de logique propositionnelle. L'ontologie est représentée sous forme d'un ensemble d'axiomes du type

$a \Rightarrow b$. Par exemple, l'axiome `apple` $\Rightarrow$ `fruit` de l'ontologie O signifie que toute recette contenant des pommes est une recette contenant des fruits.

Toutes les recettes du livre de recettes sont indexées par une conjonction de littéraux. Par exemple, la recette de la tarte aux pommes, que nous noterons R est indexée par :

$$Idx(R) = \texttt{apple} \wedge \texttt{pie} \wedge \texttt{sugar} \wedge \texttt{pastery}$$

L'ensemble des recettes indexées constitue la base de cas du système.

2.3.3 Rapports d'erreurs.

Nous sommes confrontés à deux principaux types d'erreurs. Le premier provient des recettes « mal écrites » et l'autre provient des concepts manquants dans l'ontologie. Dans les deux cas, les erreurs doivent être corrigées afin que l'application fonctionne au mieux. Voici quelques exemples d'erreurs.

Recettes mal écrites. Certaines parties de certaines recettes requièrent des correction car elles présentent des erreurs typographiques.

- `<IN>4 ts Baking power</IN>` n'a rien à voir avec le super pouvoir de cuire les choses mais devrait être remplacé par `<IN>4 ts Baking powder</IN>` qui, en revanche, à le pouvoir de faire lever les gâteaux ;
- les deux lignes d'ingrédients consécutives `<IN>1 lb Boneless pork, cut in 3/4</IN>` et `<IN>Inch cubes</IN>` devraient être fusionnées en une seule `<IN>1 lb Boneless pork, cut in 3/4 inch cubes</IN>`,
- `<IN>Salt ; pepper, Worcestershire and lemon juice</IN>`
  devrait être découpé en `<IN>Salt</IN>`, `<IN>pepper</IN>`, `<IN>Worcestershire</IN>` et `<IN>lemon juice</IN>`.

La plupart de ces erreurs ont été détectées par les experts alors qu'ils vérifiaient les recettes annotées.

Concepts absents de l'ontologie. Certaines lignes d'ingrédients n'ont été indexées par aucun des termes de la base terminologique. Cela signifie qu'aucun concept de type « ingrédient » n'a été reconnu dans cette ligne. Dans une telle situation, l'expert doit vérifier dans un premier temps que la ligne est écrite correctement. Si c'est le cas, l'erreur provient certainement de concepts manquants dans l'ontologie ou de termes manquants dans la base terminologique. Par exemple, si le concept `spam`[4] n'existe pas dans la hiérarchie des ingrédients, alors la ligne d'ingrédients `<IN>1/2 cn Spam, in 3/4" cubes</IN>` ne peut pas être indexée. L'expert doit alors ajouter le nouveau concept `spam` dans l'ontologie, définir sa position dans la hiérarchie, et associer ce concept à une liste de termes (*spam*,...) dans la base terminologique. Une fois toutes ces étapes effectuées, les différents éléments sont mis à jour et l'erreur ne devrait plus se reproduire lors de l'exécution du processus d'annotation suivant. Des problèmes identiques existent pour les hiérarchies de types, de moments et d'origine des plats.

2.4 Raisonnement à partir de cas

Envoyer des requêtes au système. Les requêtes exprimées en langue naturelle sont traitées grâce à une interface web et formalisées dans le système par une conjonction de littéraux. Pour cela, l'interface aide l'utilisateur à saisir la requête et à l'organiser de façon à ce qu'elle soit intelligible par le système. Pour reprendre l'exemple initial, la requête « je veux un dessert avec de la rhubarbe mais sans chocolat » est transformée en une requête notée Q telle que :

$$Q = \texttt{dessert} \wedge \texttt{rhubarb} \wedge \neg\texttt{chocolate}$$

[4] *Spam* signifie *spiced ham*, une sorte de jambon épicé en conserve.

Mécanismes de remémoration et d'adaptation. La première étape du processus de RÀPC consiste à retrouver parmi les recettes disponibles une recettes similaire à la requête. La remémoration est effectuée par le biais d'une classification s'appuyant à la fois sur la requête et sur les recettes indexées. Dans un premier temps, une classification dure est appliquée : le système recherche un recette dont l'index correspond exactement à l'index de la requête. Si la classification dure échoue (c'est-à-dire si aucune recette de la base ne permet de répondre immédiatement et sans adaptation à la requête), une classification élastique est alors déclenchée : la requête est généralisée et exécutée jusqu'à ce qu'une solution satisfaisante soit obtenue [5]. La classification élastique conduit à une mise en correspondance approximative des résultats et implique donc l'adaptation des recettes retrouvées afin de répondre à la requête.

L'adaptation d'une recette s'appuie sur des connaissances d'adaptation. Dans la première version de TAAABLE, les connaissances d'adaptation sont uniquement présentes sous forme de substitutions d'ingrédients. Par exemple, une recette de tarte aux pommes peut être adaptée en une recette de tarte à la rhubarbe en substituant les pommes par la rhubarbe dans la recette originale. Cependant, nous pourrions souhaiter effectuer des adaptations plus complexes de cette recette (comme le fait d'ajouter du sucre à la recette adaptée pour adoucir le goût de la rhubarbe). Un de nos objectifs est de rendre cela possible dans la future version de l'application.

2.5 Synthèse : leçons tirées de TAAABLE

Il est évident que la base de connaissances utilisée par le moteur de RÀPC n'est ni correcte ni complète et doit en permanence être améliorée et mise à jour. Cependant, dans TAAABLE, l'indépendance des deux parties du systèmes (*online* et *offline*) rend cette mise à jour très compliquée : rien ne peut être fait dynamiquement ni interactivement et aucune prise en compte « directe » du *feedback* n'est possible. C'est sans aucun doute une limitation importante de l'architecture de TAAABLE. D'un point de vue purement pratique, c'était également une limitation forte pendant le développement de l'application : la moindre modification d'un des éléments de la base de connaissances demandait un redéploiement totale de l'application, ce qui est vite devenu ingérable et a généré beaucoup de problèmes qui auraient pu être évités.

Pour tenter de résoudre ces problèmes, nous avons décidé de lier les deux parties dans une nouvelle version de TAAABLE (que nous avons baptisée WIKITAAABLE). Dans WIKITAAABLE, le moteur de RÀPC et les outils de gestion des connaissances sont couplés et travaillent de concert. Les premiers effets bénéfiques sur l'évolution de la base de connaissances et la correction des erreurs sont déjà visibles au sein du groupe de travail.

Pourquoi avons nous choisi Semantic Media Wiki ? Des cas d'utilisation pratiques des wikis sémantiques ont déjà été présentés dans la littérature [4]. Cependant, dans notre projet, nous nous devons d'aller plus loin que ce qui est fait dans ces projets. Les simples outils de navigation et d'édition des bases de connaissances, ne sont pas suffisants car nous devons également intégrer un moteur de raisonnement à l'application et faire en sorte de ré-injecter directement le *feedback* des utilisateurs dans le système sans passer par des intermédiaires coûteux. En résumé, ce que nous devons faire c'est un outil permettant une intégration souple des interactions humain-humain et humain-machine. Les wiki semblent être des outils appropriés pour cela. Le wiki est un outil social destiné à apporter un support aux collaborations entre humains. Le wiki sémantique apporte l'avantage supplémentaire de permettre la construction collaborative de connaissances en impliquant à la fois les humains et les machines. Nous pensons que les wiki sémantiques sont donc des outils particulièrement prometteurs dans une perspective de partenariat humain-machine.

Avant de porter notre choix sur Semantic Media Wiki, nous avons étudié certaines alternatives. Par exemple, l'approche utilisée dans IkeWiki [7] est bien adaptée à nos besoins puisqu'elle s'appuie sur une ontologie de fond qui est pré-chargée dans la base de connaissances et qu'elle limite l'indexation des éléments aux prédicats définis dans la base de connaissances. L'approche utilisée dans Semantic

Media Wiki est également intéressante puisque les données et leurs sémantique rendent le contenu du wiki exploitable par des processus automatiques et que ces données sont noyées dans le texte du wiki et donc visibles par les utilisateurs sans problème.

Nous avons donc décidé d'utiliser Semantic Media Wiki pour plusieurs raisons. Tout d'abord, Semantic Media Wiki est basé sur Media Wiki, et utilise donc la même interface. Or, l'interface de Media Wiki est très populaire puisque ce wiki est utilisé par des sites mondialement connus comme : le site de la communauté RÀPC francophone[5] le site de la communauté CBR internationale[6] le site, moins connu, appelé wikipédia[7]. Par conséquent, nous pensons que la plupart des utilisateurs familiers de Media Wiki seront en mesure de s'approprier facilement l'interface. Par ailleurs, la forte communauté de développeurs autour des projets Media Wiki et Semantic Media Wiki nous apporte l'assurance de disposer d'outils fiables et évoluant rapidement.

Dans la section suivante, nous présentons WIKITAAABLE et nous montrons comment nous y avons intégré les différents éléments de TAAABLE : les bases de connaissances, le moteur de raisonnement à partir de cas et l'interface.

3 WIKITAAABLE : un wiki sémantique pour TAAABLE

Avec WIKITAAABLE, la nouvelle génération du système TAAABLE, nous nous attaquons à plusieurs problèmes que présentait l'application en utilisant Semantic Media Wiki (SMW) [8] comme un tableau noir pour la construction des connaissances. Le wiki sémantique nous apporte des solutions techniques simples à mettre en œuvre pour parcourir, éditer, valider et interroger la base de connaissances représentée dans le wiki. Des travaux dans ce sens ont été illustrés dans [4]. De plus, grâce à ce wiki, la base de connaissances peut être automatiquement mise à jour par les résultats produits par le moteur de RÀPC et par l'outil d'annotation automatique. La vue de l'architecture du système est présentée figure 4 et les différents composant de cette architecture sont détaillés ci-dessous.

Semantic Media Wiki. Semantic Media Wiki est utilisé comme un tableau noir partagé entre les utilisateurs, le moteur de raisonnement à partir de cas et les bots d'annotation et d'indexation des recettes (un bot est un petit programme chargé d'exécuter une tâche automatique). Le moteur et les bots s'appuient sur un ensemble de requêtes sémantiques prédéfinies pour collecter leurs entrées. La base de connaissances est représentée par un réseau sémantique de pages wiki. Les pages représentent les recettes indexées, l'ontologie des ingrédients et l'ontologie des types de plats (voir figure 5).

Interface web utilisateur Media Wiki. C'est l'interface par défaut de l'application. Grâce à cette interface, les utilisateurs peuvent ajouter de nouvelles recettes, modifier les ingrédients, les types de plats et les origines des plats.

Bot d'annotation des recettes. Le bot d'annotation des recettes parcourt les pages contenant des recettes, en extrait les informations sur les ingrédients et met à jour les pages de recettes avec les annotations sémantiques permettant l'indexation et la catégorisation des recettes. La navigation entre les recettes et leur mise à jour est effectuée grâce à l'API de Media Wiki et l'accès à la base de connaissances se fait par l'intermédiaire de requêtes sémantiques prédéfinies. La figure 6 montre une recette originale et la visualisation du résultat de son indexation par le bot dans Semantic Media Wiki. Les relations naires de Semantic Media Wiki sont utilisées pour représenter les différents éléments d'une ligne d'ingrédients.

RÀPC. Le moteur de RÀPC récupère la base de connaissances grâce à un ensemble de requêtes sémantiques prédéfinies. Ensuite, il répond aux requêtes émises depuis l'interface web en appliquant les principes du RÀPC. Les recettes proposées au travers de l'interface sont des recettes originales ou bien des adaptations de recettes existantes. Les utilisateurs sont invités à donner un

[5] http://liris.cnrs.fr/rapc
[6] http://cbrwiki.fdi.ucm.es
[7] http://www.wikipedia.org

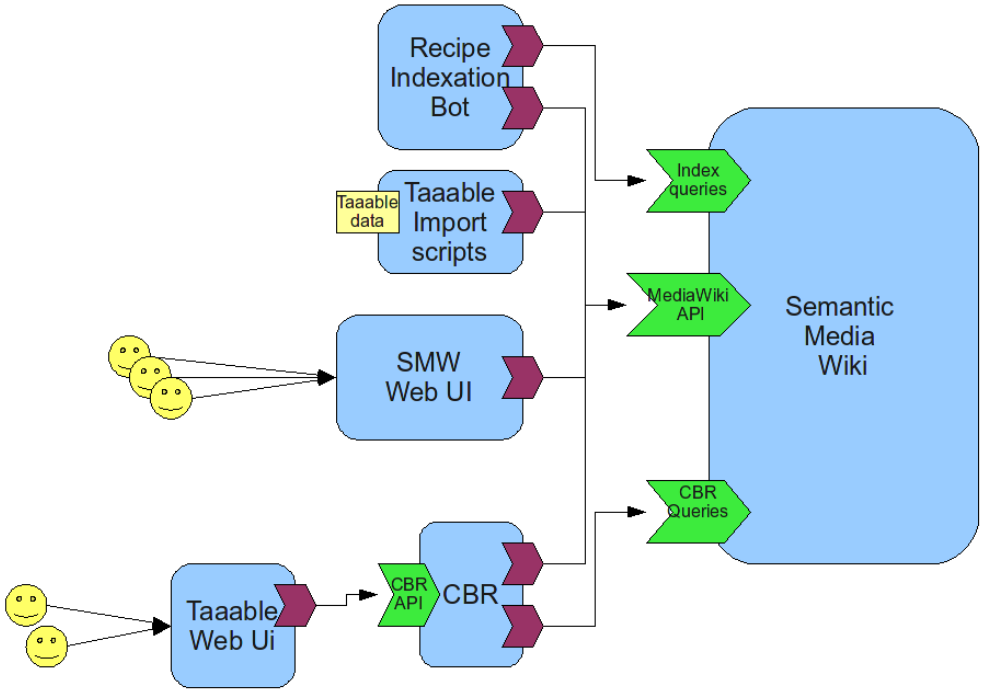


FIG. 4 – Les composants de WIKITAAABLE.

retour sur les différentes recettes. La figure 7 illustre une façon possible de capturer facilement les retours des utilisateurs. La capture d'écran n'est qu'une maquette et l'interface est en cours de développement. Cette figure illustre le scénario suivant. Supposons qu'un utilisateur souhaite une « *pie* » avec des « *oranges* » avec dans l'idée d'obtenir une tarte à l'orange. Le système propose alors les résultats présentés dans la figure 7. L'utilisateur choisit alors « *I don't like it* » pour la première proposition car il estime que remplacer le jus de citron par des oranges dans la recette « *apple crumble pie* » ne permet pas de transformer une « *apple crumble pie* » en une « *orange pie* ». En revanche, les proposition 2 et 3 sont acceptables. Ainsi, l'utilisateur choisi « *I like it* » pour ces deux propositions.

Si le *feedback* est positif, la nouvelle recette est ajoutée à la base de connaissances et une nouvelle page de wiki sémantique est créée pour la représenter. La nouvelle recette est marquée comme étant « générée » afin de ne pas être confondue avec les recettes du livre de recettes initial. Si le feedback de l'utilisateur est négatif, la recette générée est également ajoutée à la base de connaissances et une nouvelle page est créée. Toutefois, cette nouvelle page est rangée dans la catégorie « refusée ». Les recettes refusées sont gardées pour des usages futurs. Par exemple, les experts peuvent de temps à autres analyser les recettes refusées et les utiliser comme une base de travail pour un processus d'acquisition de connaissances dirigé par les échecs [3].

Scripts d'import de TAAABLE. Nous avons écrit des scripts permettant d'importer la base de connaissances actuelle de TAAABLE dans Semantic Media Wiki. La base de recettes contient environ 1400 recettes, l'ontologie des ingrédients contient 8506 classes différentes d'ingrédients. Enfin, nous utilisons *RAP - RDF API for PHP* pour explorer l'ontologie.

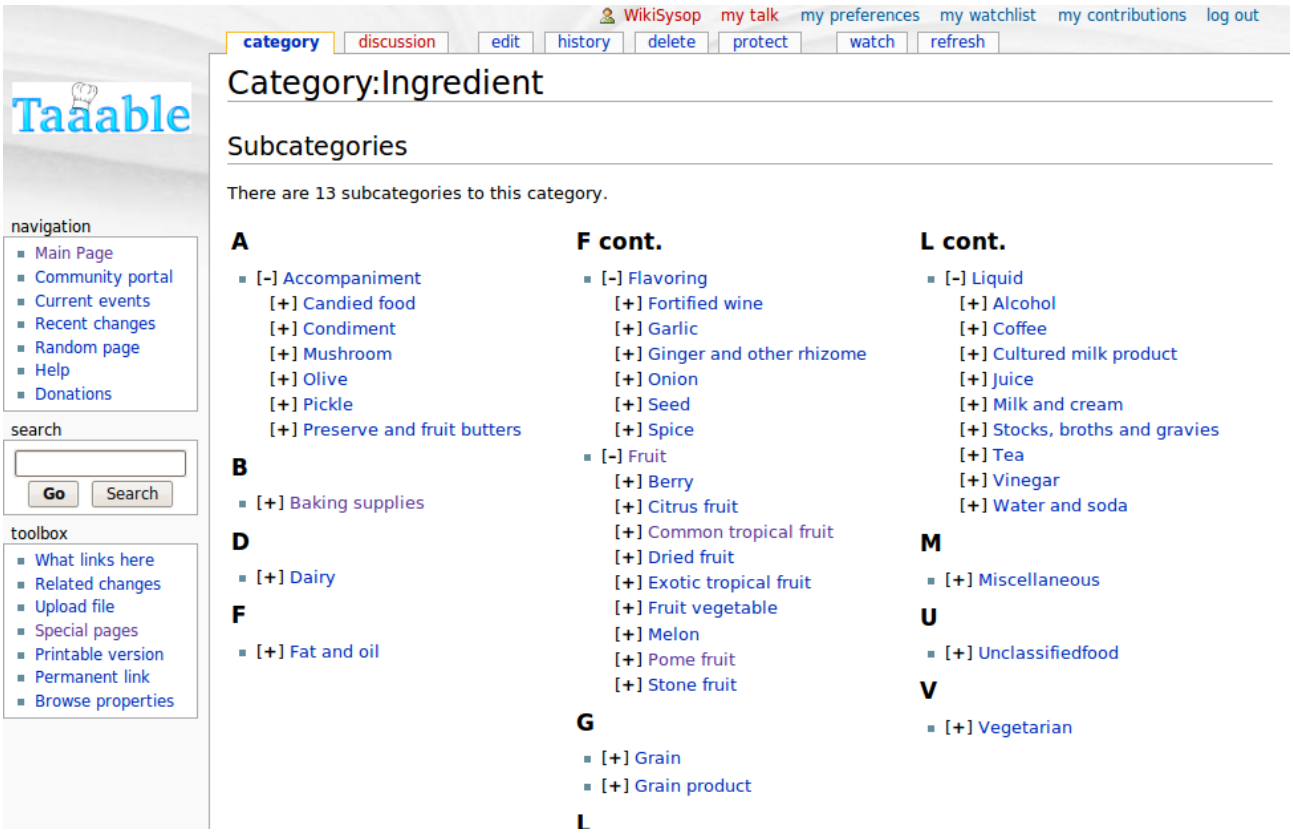


FIG. 5 – L'ontologie des ingrédients représentée sous WIKITAAABLE.

4 Discussion et travaux présents et futurs

Comme nous l'avons vu, les wiki sémantiques permettent à une communauté de mettre à jour et d'enrichir facilement un ensemble de recettes annotées ainsi qu'une ontologie. Le système WIKITAAABLE bénéficie de ces atouts et a de nombreux avantages par rapport à la version originale de TAAABLE.

- Les utilisateurs peuvent ajouter de nouvelles recettes facilement.
- Les utilisateurs peuvent corriger des recettes mal annotées.
- Les utilisateurs peuvent naviguer et parcourir l'ontologie. simplement.
- Les personnes responsables de la maintenance de l'ontologie peuvent rapidement modifier l'ontologie des ingrédients, des types de plats et tester les effets des modifications sur le moteur de raisonnement.
- Le *feedback* sur l'adaptation des recettes peut désormais être capturé et représenté dans les wiki sémantiques. La base de connaissances du système augmente du simple fait d'être utilisée.

Une des raisons du succès des systèmes de RÀPC est qu'ils sont théoriquement capables d'apprendre à partir de l'expérience en acquérant des connaissances additionnelles après chaque épisode de résolution de problème. Dans TAAABLE cependant, apprendre à partir de l'expérience est difficile en raison du manque de mécanismes embarqués permettant la prise en compte du *feedback* pour améliorer les bases de connaissances existantes. Dans WIKITAAABLE, l'utilisation du wiki sémantique facilite ce processus en permettant l'enrichissement et la mise à jour des données et des connaissances utilisées par le système par une communauté d'utilisateurs. Par exemple, les mainteneurs des ontologies peuvent facilement les modifier et tester l'impact de ces modifications sur les résultats produits par le moteur de RÀPC.

Le développement de WIKITAAABLE a soulevé de nombreuses question qui sont principalement liées au maintien de la cohérence du système. Comment peut-on garantir la cohérence d'une application alors que plusieurs utilisateurs ayant souvent des points de vues différents, sont autorisés à modifier

```
== Ingredients ==
*   1 c rice
*   2 c water
*   1/2 c sugar
*   1 ts salt
*   2 c evaporated milk
*   1 c raisins
*   3 eggs separated
*   3/4 ts vanilla
*   1/4 ts cinnamon
*   1/4 ts nutmeg

== Preparation ==

* combine the rice water sugar and salt in a large saucepan bring the
 water to a boil and cover the saucepan reduce the heat to low and
 continue to cook for 12 - 15 minutes or until the water is absorbed
 combine the milk and egg yolk add them to the rice then mix in the
 raisin vanilla and cinnamon simmer for five minutes remove from the
 heat beat the egg white until stiff fold them into the rice chill and
 garnish with nutmeg before serving also taste good warm
```

FIG. 6 – Un exemple de recette annotée : « ARROZ DULCE ».

les connaissances codées dans le système ? Comment peut-on combiner efficacement une procédure semi-automatique et un processus d'enrichissement manuel pour construire une ontologie ? Ces questions ont une importance capitale puisque l'ontologie joue un rôle central dans le système TAAABLE et est utilisées à deux niveaux principaux (elle guide le processus d'indexation en identifiant les concepts impliqués dans chaque recette et elle est utilisée par le système de RÀPC pour adapter les recettes).

Si n'importe quel utilisateur peut librement modifier l'ontologie, le risque d'introduire des incohérences dans les connaissances est d'autant plus élevé. En conséquence, selon une telle hypothèse,

FIG. 7 – Capturer le *feedback* utilisateur grâce à l'interface de TAAABLE.

le moteur de RÀPC et les bots d'indexation et d'annotation produiront certainement des résultats imprévisibles. Plusieurs stratégies peuvent être imaginées pour éviter ce problème :

- Restreindre les droits de mise à jour de l'ontologie aux « administrateurs ». C'est une solution possible au problème, mais c'est également une limitation considérable au travail collaboratif. De plus, l'amélioration de l'ontologie dépend fortement de la disponibilité des dits administrateurs et risque donc de ne pas être faite souvent.
- Validation des changements avant qu'ils ne soient intégrés à la version en production du système. Cette stratégie a également quelques limites : le support de tels processus n'est pas trivial dans les wikis sémantiques (car ils ne sont pas conçus pour cela), les problèmes de synchronisation entre plusieurs versions d'un système unique sont important, et pour finir, tout cela est très coûteux en temps et pas très motivant pour les administrateurs. De plus, cela ne résout pas le problème des conflits entre plusieurs changements en parallèle de l'ontologie.
- Adaptation d'approches d'intégration continue utilisées en génie logiciel dans le contexte du système WIKITAAABLE. Par exemple, si l'adaptation d'une recette a été validée par plusieurs utilisateurs, alors l'ontologie devrait être modifiée afin de préserver cette adaptation (dans ce cas, le *feedback* des utilisateurs collecté par WIKITAAABLE devrait générer des tests de non-régression).
- Vers un WIKITAAABLE pair à pair ? Dans une telle approche, chaque utilisateur pourrait avoir sa propre version de l'application s'appuyant à la fois sur une base de connaissances commune et sur une base de connaissances d'adaptation personnelle. Les connaissances d'adaptation pourraient alors être partagées entre plusieurs utilisateurs sur la base de la confiance qu'ils accordent à leurs pairs.

La prochaine étape du développement de WIKITAAABLE est d'intégrer totalement le moteur de RÀPC au wiki sémantique et de mettre en place des possibilités d'interactions à plusieurs niveaux. Une attention toute particulière sera portée à la capacité du wiki sémantique à représenter et gérer des règles d'adaptation complexes incluant, par exemple, des contraintes booléennes.

Références

[1] F. Badra, R. Bendaoud, R. Bentebitel, P.A. Champin, J. Cojan, A. Cordier, S. Desprès, S. Jean-Daubias, J. Lieber, T. Meilender, A. Mille, E. Nauer, A. Napoli, et Y. Toussaint. TAAABLE : Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In M. Schaaf, editor, *ECCBR 2008, The 9th European Conference on Case-Based Reasoning, Trier, Germany, September 1-4, 2008, Workshop Proceedings*, pages 219–228, 2008.

[2] Michel Buffa, Fabien L. Gandon, Guillaume Ereteo, Peter Sander, et Catherine Faron. Sweetwiki : A semantic wiki. *Journal of Web Semantics*, 6(1) :84–97, 2008.

[3] Améie Cordier. *Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning*. PhD Thesis, Université Claude Bernard Lyon 1, November 2008.

[4] Markus Krötzsch, Sebastian Schaffert, et Denny Vrandecic. Reasoning in semantic wikis. In Grigoris Antoniou, Uwe Aßmann, Cristina Baroglio, Stefan Decker, Nicola Henze, Paula-Lavinia Patranjan, et Robert Tolksdorf, editors, *Reasoning Web*, volume 4636 of *Lecture Notes in Computer Science*, pages 310–329. Springer, 2007.

[5] J. Lieber. Strong, Fuzzy and Smooth Hierarchical Classification for Case-Based Problem Solving. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-02), Lyon, France*, pages 81–85. IOS Press, Amsterdam, 2002.

[6] C. K. Riesbeck et R. C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[7] S. Schaffert. IkeWiki : A Semantic Wiki for Collaborative Knowledge Management. *1st International Workshop on Semantic Technologies in Collaborative Applications (STICA06), Manchester, UK*, 2006.

[8] Max Völkel, Markus Krötzsch, Denny Vrandecic, Heiko Haller, et Rudi Studer. Semantic wikipedia. *Journal of Web Semantics*, 5(4), 2007.

Le moteur de raisonnement à partir de cas de WIKITAAABLE

Comment l'approche de la remémoration-adaptation s'appuyant sur des reformulations peut s'appliquer sur une base de cas relativement volumineuse et peu organisée

Jean Lieber
Équipe Orpailleur,
LORIA, UMR 7503 CNRS, INRIA, Nancy Université
BP 239 54506 Vandœuvre-lès-Nancy, France, Jean.Lieber@loria.fr

Résumé

Le système WIKITAAABLE est un système de raisonnement à partir de cas dont les cas sont des recettes de cuisine et les problèmes des requêtes données par des ingredients et types de plats requis ou à éviter. Son moteur d'inférences s'appuie sur le modèle des reformulations et des chemins de similarité : la remémoration met en évidence une recette et un chemin de similarité qui réifie le lien recette-requête et sert de base au processus d'adaptation. Si la base de recettes était organisée de manière adéquate, la remémoration pourrait être mise en œuvre grâce à des modifications à la fois sur la base de recettes (*via* cette organisation) et sur la requête, dans l'optique d'apparier exactement au moins une recette modifiée et la requête modifiée (le chemin serait alors décrit par la séquence des modifications). Dans la version actuelle, le travail d'organisation de la base de recettes n'ayant pas été fait (est-il seulement faisable ?), la remémoration consiste en la modification seule de la requête. Une fois une recette remémorée, elle peut être adaptée selon cette première version du chemin de similarité mais un nouvel appariement avec modifications de la recette et de la requête peut éventuellement aboutir à un « meilleur » chemin de similarité, i.e., un chemin conduisant à une adaptation moins risquée. Cette notion de risque est mesurée à l'aide d'une fonction de coût associée aux chemins de similarité. La représentation actuelle des connaissances utilisées par le système est limitée ; elle ne permet pas de représenter, par exemple, des quantités. Une réflexion sur ce que devrait être la nouvelle représentation est également menée dans cet article.

Mots clés : WIKITAAABLE, reformulation, chemin de similarité, remémoration, adaptation

1 Introduction

Un cas est un élément de connaissance représentant une expérience de résolution de problème. Effectuer un raisonnement à partir de cas (RÀPC [20]), c'est résoudre des problèmes en faisant appel à des cas. Les notions de problème et de solution sont supposées dépendantes du domaine d'application. Cela consiste souvent en une remémoration (choix d'un cas jugé similaire au problème cible) suivie d'une adaptation du cas remémoré en vue de la résolution du problème cible. Une grande partie des recherches qui se posent dans ce domaine est « Comment développer un système de RÀPC ? » Cette question se pose de façon pratique dans le cadre du projet TAAABLE, dont l'objectif est de proposer un système pour participer au *Computer Cooking Contest* (CCC [1]). Ce projet a donné lieu au premier système TAAABLE, qui a été classé deuxième au premier CCC, en 2008 [1]. Le deuxième système TAAABLE étend le premier de plusieurs manières. D'abord, il intègre des principes et technologies des wikis sémantiques, ce qui lui vaut d'être baptisé du beau nom de WIKITAAABLE [8]. Ensuite, il intègre de nouveaux travaux sur l'acquisition et l'extraction des connaissances (en particulier [3, 2] et [7]). Enfin, quelques améliorations sur le moteur de RÀPC du premier système TAAABLE ont été effectuées. Cet article est l'occasion pour décrire ce moteur en détail.

[1] Un compétiteur du CCC doit faire un système pour résoudre des problèmes de cuisine (requête : des ingrédients et types de plats autorisés ou interdits ; résultat : une ou plusieurs recettes satisfaisant la requête) en s'appuyant sur un livre de cuisines, qui fait office de base de cas. Le site du concours de 2009 est http://www.wi2.uni-trier.de/ccc09/index.php.

Une approche pour implanter un système de RÀPC s'appuie sur la notion de reformulation. Cette approche est rappelée à la section 2. Son application directe à WIKITAAABLE pose un problème : la base de cas (ou livre de recettes) étant peu structurée et relativement volumineuse, la remémoration prend souvent trop de temps de calcul et de place mémoire. Du coup, cette approche a été adaptée : La section 3 décrit brièvement l'application WIKITAAABLE et zoome sur la base de connaissances de ce système. Le moteur de ce système s'appuie sur la philosophie de la remémoration guidée par l'adaptation (*adaptation-guided retrieval* [21]) : tout cas remémoré doit être adaptable. C'est ce qui explique que la description de l'adaptation (section 4) précède celle de la remémoration (section 5). La remémoration et l'adaptation s'appuient sur diverses connaissances, dont certaines sont numériques : ce sont les coûts des reformulations. La section 6 est une discussion sur le choix de ces coûts. La section 7 présente quelques réflexions sur les changements à venir au niveau du formalisme de la base de connaissances et sur les modifications du moteur que ces changements vont entraîner. La section 8 conclut cet article.

2 Rappels sur le RÀPC et les reformulations

Rappels. Raisonner à partir de cas, c'est résoudre des problèmes en faisant appel à une base de cas, un cas étant la représentation d'un épisode de résolution de problème. Un cas source est un cas de la base de cas. On considère souvent qu'il est donné par un couple $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ où $\texttt{srce}$ est un problème et où $\texttt{Sol}(\texttt{srce})$ est une solution de $\texttt{srce}$. Soit $\texttt{cible}$, le problème à résoudre (où problème cible) lors d'une session de RÀPC particulière. Cette session est souvent constituée d'une *remémoration* (sélection d'un cas source $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ jugé similaire à $\texttt{cible}$) et d'une *adaptation* de $\texttt{Sol}(\texttt{srce})$ en une solution (candidate) $\texttt{Sol}(\texttt{cible})$ de $\texttt{cible}$.

Une *reformulation* est un couple $(\texttt{r}, \mathcal{A}_\texttt{r})$ où $\texttt{r}$ est une relation binaire sur l'espace des problèmes et où $\mathcal{A}_\texttt{r}$ est une application qui à un triplet $(\texttt{srce}, \texttt{Sol}(\texttt{srce}), \texttt{cible})$ tel que $\texttt{srce}\ \texttt{r}\ \texttt{cible}$ associe $\texttt{Sol}(\texttt{cible})$, une solution de $\texttt{cible}$. Autrement dit, $\mathcal{A}_\texttt{r}$ est une fonction d'adaptation valable dans le contexte particulier des *problèmes d'adaptation* $(\texttt{srce}, \texttt{Sol}(\texttt{srce}), \texttt{cible})$ tels que les deux problèmes, $\texttt{srce}$ et $\texttt{cible}$, sont liés par $\texttt{r}$. Une reformulation constitue donc une connaissance d'adaptation[2].

Un *chemin de similarité* d'un problème $\texttt{srce}$ vers un problème $\texttt{cible}$ est un ensemble d'assertions $\texttt{pb}_{i-1}\ \texttt{r}_i\ \texttt{pb}_i$, $1 \leqslant i \leqslant n$, où $(\texttt{r}_i, \mathcal{A}_{\texttt{r}_i})$ est une reformulation, les $\texttt{pb}_i$ sont des problèmes, $\texttt{pb}_0 = \texttt{srce}$ et $\texttt{pb}_n = \texttt{cible}$. On le note

$$\texttt{srce} = \texttt{pb}_0\ \texttt{r}_1\ \texttt{pb}_1 \ldots \texttt{pb}_{n-1}\ \texttt{r}_n\ \texttt{pb}_n = \texttt{cible}$$

Si on a établi un chemin de similarité de $\texttt{srce}$ à $\texttt{cible}$, on peut adapter $\texttt{Sol}(\texttt{srce}) = \texttt{Sol}(\texttt{pb}_0)$ en une solution $\texttt{Sol}(\texttt{pb}_n) = \texttt{Sol}(\texttt{cible})$ de $\texttt{cible}$ en appliquant successivement $\mathcal{A}_{\texttt{r}_1}$, $\mathcal{A}_{\texttt{r}_2}$, ..., $\mathcal{A}_{\texttt{r}_n}$. Cette séquence s'appelle *chemin d'adaptation*.

D'un point de vue algorithmique, la recherche d'un chemin de similarité de longueur $n \geqslant 2$ suppose le test de la composition de deux relations binaires. Or ce test n'est pas nécessairement décidable, même si les deux relations le sont. Cela justifie le fait de se restreindre à une famille de chemins de similarité dont la découverte soit décidable. Tout d'abord, soit $\sqsupseteq$ une relation entre problèmes telle que si $\texttt{pb} \sqsupseteq \texttt{pb}'$ et que $\texttt{Sol}(\texttt{pb})$ est une solution de $\texttt{pb}$, alors $\texttt{Sol}(\texttt{pb})$ est aussi une solution de $\texttt{pb}'$ ($\sqsupseteq$ est une relation de généralité entre problèmes [17]). Ainsi, $(\sqsupseteq, \mathcal{A}_\sqsupseteq)$ est une reformulation si on pose $\mathcal{A}_\sqsupseteq(\texttt{srce}, \texttt{Sol}(\texttt{srce}), \texttt{cible}) = \texttt{Sol}(\texttt{srce})$ (adaptation par copie). Par ailleurs, on définit la notion d'opérateur sur les problèmes ainsi : un opérateur op est une fonction qui à un problème $\texttt{pb}$ associe un ensemble fini (et éventuellement vide) de problèmes. Si $\texttt{pb}' \in op(\texttt{pb})$, on note $\texttt{pb} \xrightarrow{op} \texttt{pb}'$. Si $\xrightarrow{op}$ est une relation fonctionnelle, on notera $\texttt{pb} \xmapsto{op} \texttt{pb}'$ et $\texttt{pb}' = op(\texttt{pb})$. On suppose ensuite que les reformulations disponibles en dehors de $(\sqsupseteq, \mathcal{A}_\sqsupseteq)$ sont de la forme $(\xrightarrow{\sigma}, \mathcal{A}_{\xrightarrow{\sigma}})$ ou de la forme $(\xleftarrow{\gamma}, \mathcal{A}_{\xleftarrow{\gamma}})$. Enfin, on supposera que les chemins de similarité sont de la forme

$$\texttt{srce} \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_p} \Sigma(\texttt{srce}) \sqsupseteq \Gamma(\texttt{cible}) \xleftarrow{\gamma_q} \ldots \xleftarrow{\gamma_1} \texttt{cible}$$

[2]Le terme reformulation est très discutable, mais a été conservé pour des raisons historiques [18]. On aurait également pu choisir le terme « opérateur d'adaptation » qui est utilisé dans [7].

Ce qui conduit à une adaptation appliquant successivement $\mathcal{A}\,\underrightarrow{\sigma_1}$, ..., $\mathcal{A}\,\underrightarrow{\sigma_p}$, $\mathcal{A}\,\underleftarrow{\gamma_q}$, ... et $\mathcal{A}\,\underleftarrow{\gamma_1}$ ($\mathcal{A}_\sqsupseteq$ étant une copie est inutile). Établir un chemin de similarité de cette forme étant donné `srce` et `cible` peut se faire par une exploration dans un espace d'états utilisant l'algorithme A*. Le problème est que pour la remémoration, seul `cible` est connu. En général, tenter d'apparier l'un après l'autre les cas sources est trop coûteux (surtout pour une grande base de cas). Dans ce cas, on peut envisager (au moins) deux solutions. La première a été mise au point pour l'application RÉSYN/RÀPC [14] et consiste à utiliser une organisation hiérarchique des cas sources. Pour dire les choses rapidement, cette organisation s'appuie sur une factorisation des cas sources sous la forme de subsumants communs et l'accès aux cas sources se fait via cette hiérarchie. Cela fait qu'au lieu de considérer chaque cas source, on considère les plus proches d'abord. Cette approche suppose une organisation adéquate de la base de cas. L'autre solution est celle décrite dans cet article : elle consiste à n'appliquer, lors de la remémoration, que des modifications sur `cible`, ce qui sera efficace si la recherche exacte des `srce` tels que $\texttt{srce} \sqsupseteq \Gamma(\texttt{cible})$ est rapide. Un avantage de cette dernière approche est qu'elle est plus rapide à mettre en œuvre. Son inconvénient majeur tient au fait que le chemin de similarité établi lors de la remémoration n'est pas nécessairement optimal ; on peut alors le recalculer en appliquant les modifications non seulement sur `cible` mais aussi sur `srce`, maintenant que $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ a été sélectionné par la remémoration.

Application à WIKITAAABLE. Ce modèle du RÀPC et des reformulations peut s'appliquer à WIKITAAABLE moyennant certaines adaptations.

Dans l'application WIKITAAABLE, le livre de recettes est la base de cas. Or, un cas $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ représente un épisode de résolution de problème : `srce` a pour solution $\texttt{Sol}(\texttt{srce})$. Pour appliquer ces définitions à une recette R, il faut pouvoir l'écrire comme un couple $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$. La recette constitue la solution attendue, donc $\texttt{Sol}(\texttt{srce}) = R$, mais que vaut `srce` ? Dans le cadre de cette application, un problème est une requête et, en particulier, le problème cible est une requête dénotée par Q dans la suite : $\texttt{cible} = Q$ et une solution de `cible` est une recette satisfaisant Q. `srce` est donc une des requêtes que $R = \texttt{Sol}(\texttt{srce})$ satisfait. On supposera que le langage des requêtes contient le langage des recettes[3], donc R est aussi une requête et elle est satisfaite par la recette R ; c'est de plus la requête la plus spécifique que R satisfait et on la choisit pour cette raison : $\texttt{srce} = R$([4]). Ainsi, le cas source est $(\texttt{srce}, \texttt{Sol}(\texttt{srce})) = (R, R)$. Dans ces conditions, la distinction problème-solution n'a guère d'intérêt, l'espace des solutions (les recettes) étant inclus dans celui des problèmes (les requêtes). Cette distinction ne sera plus faite dans la suite de l'article, mais on continuera de parler de recettes (les cas sources et leurs adaptations) et de requêtes.

De la même façon, les reformulations ne seront plus donnés par des couples $(\texttt{r}, \mathcal{A}_\texttt{r})$ mais par un seul élément s correspondant à une opération de transformation de requête (en particulier, de recette) : $(\texttt{r}, \mathcal{A}_\texttt{r}) = (s, s)$ dénoté simplement par s.

3 WIKITAAABLE et ses connaissances

Les connaissances de WIKITAAABLE sont principalement exprimées en logique propositionnelle. La base de connaissances de TAAABLE est un ensemble de « conteneurs de connaissances[5] » :

$$\texttt{BC} = \{\mathcal{O}, \texttt{Recettes}, \mathcal{H}_{idx}, \texttt{CA}, \texttt{coût}\}$$

`BC` est encodée dans des pages wiki et le moteur de RÀPC accède à ces pages via des requêtes SPARQL.

$\mathcal{O}$ est l'ontologie de domaine et est représentée par un ensemble d'axiomes de la forme $a \Rightarrow b$ où a et b sont des variables représentant des classes de recettes (on utilisera indifféremment les termes « variable propositionnelle » et « classe », dans la suite, quand le contexte ne sera pas ambigü). Par exemple, `citron` (resp., `agrume`)

[3] Ce qui ne sera vrai qu'au niveau des représentations formelles $idx(R)$, voir section 3.

[4] Un autre découpage d'une recette R en un cas $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ avec $\texttt{Sol}(\texttt{srce}) = R$ consiste à choisir `srce` dynamiquement, lors de la remémoration : ce sera la modification $\Gamma(Q) = \Gamma(\texttt{cible})$ de la requête qui donne la recette. Ce n'est cependant pas l'option que nous avons choisie ici.

[5] Un conteneur de connaissances (*knowledge container* [19]) est une base de connaissances spécifique à un certain type de connaissances (ontologie, cas, etc.) et aux inférences associées. La base de connaissances d'un système de RÀPC est un ensemble de conteneurs de connaissances et de liens entre ces conteneurs (p. ex., les cas sont exprimés à l'aide de concepts de l'ontologie).

représente la classe des recettes avec des citrons (resp., avec des agrumes) et l'axiome $\texttt{citron} \Rightarrow \texttt{agrume}$ exprime le fait que toute recette avec des citrons est une recette avec des agrumes. En fait, chaque nom d'ingrédient X tel que `citron` est interprété comme « La classe des recettes contenant l'ingrédient X ». Un autre exemple est donné par l'axiome $\texttt{huile\_olive} \Rightarrow \texttt{Méditerranée}$. Ici, `huile_olive` représente la classe des recettes avec de l'huile d'olive et `Méditerranée`, la classe des recettes méditerranéennes. Cet axiome affirme que toute recette avec de l'huile d'olives est d'origine méditerranéenne (si vous mettez une goutte d'huile d'olive dans votre bœuf Strogonoff, il devient méditerranéen).

$\mathcal{O}$ est (représentée par) une hiérarchie dont les arcs (a, b) représentent les axiomes $a \Rightarrow b$. $\top$ est la racine de cette hiérarchie et dénote l'univers des recettes.

`Recettes` est l'ensemble des recettes données par les organisateurs du CCC, et est par conséquent, la base de cas du système de RÀPC WIKITAAABLE [6]. Une recette $R \in \texttt{Recettes}$ ne peut pas être directement exploitée par le moteur de RÀPC : celui-ci requiert une représentation formelle, alors que R est, pour sa plus grande partie, exprimée en langue naturelle (avec une petite structuration XML). Par conséquent, seule la partie formalisée de la recette R est manipulée par le moteur d'inférence, à savoir son index $idx(R)$, exprimé sous la forme d'une conjonction de littéraux[7] (le processus d'indexation est en fait un processus d'annotation de textes décrit dans [1]). Par exemple

$$idx(R) = \texttt{laitue} \wedge \texttt{vinaigre} \wedge \texttt{huile\_olive} \wedge \texttt{tomate} \wedge \textit{Rien d'autre} \quad (1)$$

est une représentation formelle et abstraite[8] de la recette R dont les ingrédients sont une laitue, du vinaigre, de l'huile d'olive et des tomates. Une hypothèse du monde clos est associée à $idx(R)$: si une propriété ne peut être déduite ni de la liste des propriétés ni de l'ontologie, elle est considérée comme étant fausse. Formellement, si $idx(R) \not\models_{\mathcal{O}} a$ alors le terme « *Rien d'autre* » de (1) est une conjonction de littéraux contenant le littéral $\neg a$.[9] Par exemple, cette hypothèse du monde clos permet de déduire que $idx(R) \models_{\mathcal{O}} \neg\texttt{viande} \wedge \neg\texttt{poisson}$, i.e., que R est une recette végétarienne[10].

On peut noter que $idx(R)$ a exactement un modèle (pour toute variable propositionnelle a, soit $idx(R) \models a$, soit $idx(R) \models \neg a$). On peut pour cette raison le considérer comme une instance de recette (une expérience particulière).

D'un point de vue algorithmique, la remémoration s'appuie sur l'algorithme de classification élastique décrit dans [16] et qui combine un parcours de la hiérarchie $\mathcal{H}_{idx}$ et une recherche A*.

$\mathcal{H}_{idx}$. Les index $idx(R)$ sont utilisés pour accéder aux recettes via la hiérarchie $\mathcal{H}_{idx}$, selon l'ordre partiel $\models_{\mathcal{O}}$: pour $C, D \in \mathcal{H}_{idx}$, $C \models_{\mathcal{O}} D$ ssi il y a un chemin de $\mathcal{H}_{idx}$ de C vers D. Les index $idx(R)$ sont les feuilles de $\mathcal{H}_{idx}$ (en effet, ce sont des minimums pour la relation d'ordre $\models_{\mathcal{O}}$ de l'ensemble des formules satisfiables).

`CA`. Les connaissances d'adaptation sont constituées de deux parties. La première est l'ontologie $\mathcal{O}$. La deuxième est un ensemble `CA` de substitutions. Toute $\sigma \in \texttt{CA}$ peut être considérée comme une règle d'inférence spécifique $\dfrac{R \text{ est une bonne recette}}{\sigma(R) \text{ est une bonne recette}}$([11]). Une substitution est donnée par deux conjonctions de

[6] Il y avait environ 900 recettes lors du premier CCC et il y en a environ 1500, pour le deuxième.

[7] Un littéral est de la forme a (littéral positif) ou de la forme $\neg a$ (littéral négatif) où a est une variable propositionnelle. Une conjonction de littéraux est une formule de la forme $\ell_1 \wedge \ldots \wedge \ell_n$ où les ℓ_i sont des littéraux.

[8] Abstraite au sens où la transformation $R \mapsto idx(R)$ s'accompagne d'une perte d'informations et d'un changement de langage (voir [4, 17]).

[9] Si f et g sont deux formules propositionnelles alors $f \models_{\mathcal{O}} g$ signifie que f entraîne g, étant donné l'ontologie $\mathcal{O}$. Plus précisément : si $\mathcal{I}$ satisfait à la fois $\mathcal{O}$ et f alors $\mathcal{I}$ doit satisfaire g.

[10] Si on définit une recette végétarienne comme étant une recette sans viande ni poisson.

[11] Cette règle d'inférence est incertaine au sens où σ ne produit pas nécessairement une bonne recette. On considérera qu'une substitution est d'autant meilleure que la production d'une mauvaise recette $\sigma(R)$ à partir d'une bonne recette R est improbable[12].

[12] Bien évidemment, ces notions de bonnes et mauvaises recettes sont subjectives, mais on fera comme si il y avait un large consensus sur le jugement de ces recettes[13].

[13] Par exemple, la tarte aux concombres, c'est pas bon.

littéraux, G et D, et est notée $G \rightsquigarrow D$ (G et D sont les parties gauche et droite de la substitution). Une substitution s'applique sur une conjonction de littéraux, par exemple sur un index de recette ou sur une requête. La substitution $G \rightsquigarrow D$ est applicable sur la conjonction de littéraux f si les littéraux de G sont inclus dans les littéraux de f ($G \supseteq f$, dans une notation ensembliste qui assimile les conjonctions de littéraux à des ensembles de littéraux). Si $\sigma = G \rightsquigarrow D$ est applicable sur f, alors, l'application de σ à f, notée $\sigma(f)$, consiste à remplacer dans f les littéraux de G par des littéraux de D ($\sigma(f) = (f \setminus G) \cup D$).

Par exemple, la substitution

$$\sigma = \texttt{salade} \wedge \neg\texttt{pomme\_de\_terre} \wedge \texttt{vinaigre} \rightsquigarrow \texttt{salade} \wedge \neg\texttt{pomme\_de\_terre} \wedge \texttt{jus\_de\_citron} \wedge \texttt{sel} \quad (2)$$

se lit ainsi : « Pour une salade qui ne contient pas de pommes de terre, s'il y a du vinaigre, on peut le remplacer par du jus de citron et du sel. » Notons que la conjonction $\texttt{salade} \wedge \neg\texttt{pomme\_de\_terre}$ apparaît en partie gauche et en partie droite de la substitution : elle représente le contexte de la substitution, i.e., des conditions pour la rendre applicable mais qui ne sont pas modifiées par la substitution.

L'*inverse* d'une substitution $\sigma = G \rightsquigarrow D$ est la substitution $\sigma^{-1} = D \rightsquigarrow G$. La *composition* de deux substitutions $\sigma_1 = G_1 \rightsquigarrow D_1$ et $\sigma_2 = G_2 \rightsquigarrow D_2$ est la substitution $\sigma = \sigma_2 \circ \sigma_1 = G \rightsquigarrow D$ avec $G = G_1 \cup (G_2 \setminus D_1)$ et $D = (D_1 \setminus G_2) \cup D_2$ (toujours en assimilant une conjonction de littéraux à l'ensemble des littéraux de la conjonction). On notera que $(\sigma_2 \circ \sigma_1)^{-1} = \sigma_1^{-1} \circ \sigma_2^{-1}$.

$\texttt{coût}$. L'inférence effectuée par le moteur de RÀPC s'appuie sur des substitutions, qui sont soit issues de $\texttt{CA}$, soit construites à l'aide de l'ontologie $\mathcal{O}$ (voir sections suivantes pour plus de détails). Le choix des substitutions à appliquer est effectué selon le contexte de résolution de problème et selon une fonction $\texttt{coût} : \sigma \mapsto \texttt{coût}(\sigma) > 0$. La substitution σ est préférée à la substitution τ si $\texttt{coût}(\sigma) < \texttt{coût}(\tau)$. Par conséquent, la fonction $\texttt{coût}$ (et ses paramètres) constitue un conteneur de connaissances supplémentaire.

4 Adaptation d'une recette

L'adaptation est composée de deux étapes. Soit R la recette à adapter (issue de la remémoration) et $idx(R)$ son index. La première étape de l'adaptation est l'appariement, dont l'objectif est de construire un chemin d'adaptation de $idx(R)$ à Q de la forme

$$idx(R) \xmapsto{\sigma_1} \ldots \xmapsto{\sigma_p} \Sigma(idx(R)) \vDash_{\mathcal{O}} \Gamma(Q) \xmapsto{\gamma_q} \ldots \xmapsto{\gamma_1} Q \quad (3)$$

où $\sigma_i \in \texttt{CA}$ ($i = 1 \ldots p$) et où les substitutions γ_j ($j = 1 \ldots q$) correspondent aux axiomes de l'ontologie : $\gamma_j = a_j \rightsquigarrow b_j$ avec $(a_j \Rightarrow b_j) \in \mathcal{O}$. Un tel chemin d'adaptation est établi selon une recherche A* dans un espace d'états et aboutit à un chemin minimisant $\sum_i \texttt{coût}(\sigma_i) + \sum_j \texttt{coût}(\gamma_j)$.

La deuxième étape de l'adaptation consiste à « suivre » le chemin d'adaptation : R est d'abord adaptée successivement en $\sigma_1(R)$, $\sigma_2(\sigma_1(R))$, ... $\sigma_p(\ldots(\sigma_2(\sigma_1(R))\ldots) = \Sigma(R)$. Puis, les ingrédients de $\Sigma(R)$ sont substitués par d'autres ingrédients selon un processus de généralisation-spécialisation (la généralisation correspond à la relation $\vDash_{\mathcal{O}}$ et la spécialisation aux substitutions γ_q^{-1}, …, γ_1^{-1}). Pour résumer, la recette R suit la chaîne de transformations suivante :

$$R \xmapsto{\sigma_1} \ldots \xmapsto{\sigma_p} \Sigma(R) \xmapsto{\gamma_q^{-1}} \ldots \xmapsto{\gamma_1^{-1}} \Gamma^{-1} \circ \Sigma(R) \quad (4)$$

Par exemple, considérons les entrées suivantes du processus d'appariement :

$$\begin{aligned} Q &= \texttt{scarole} \wedge \texttt{jus\_de\_citron} \wedge \neg\texttt{oignon} \quad (5) \\ idx(R) &= \texttt{laitue} \wedge \texttt{vinaigre} \wedge \texttt{huile\_olive} \wedge \texttt{tomate} \wedge \textit{Rien d'autre} \end{aligned}$$

Q est la requête pour une recette ayant la scarole et le jus de citron parmi ses ingrédients, mais pas d'oignon. L'appariement peut produire le chemin d'adaptation suivant (pour peu que ce soit le moins coûteux) :

$$idx(R) \xmapsto{\sigma} \Sigma(idx(R)) \vDash_{\mathcal{O}} \Gamma(Q) \xmapsto{\gamma} Q$$

où σ est définie par (2) et $\gamma = \texttt{scarole} \rightsquigarrow \texttt{salade\_verte}$. Par conséquent, l'adaptation de R consistera à remplacer le vinaigre par du jus de citron et du sel (cf. σ) et à substituer $\texttt{laitue}$ par $\texttt{scarole}$ (cf. $\vDash_{\mathcal{O}}$ et γ^{-1}).

5 Remémoration de recettes

Soit Q une requête. Par exemple, celle de l'équation (5). La remémoration vise à choisir les index $idx(R)$ s'appariant avec la requête Q. Cet appariement est exact si $idx(R) \models_{\mathcal{O}} Q$. Si aucun index ne s'apparie exactement avec Q, la requête Q est progressivement relâchée en $\Gamma(Q)$ tel qu'il existe au moins un $idx(R)$ vérifiant $idx(R) \models_{\mathcal{O}} \Gamma(Q)$. Le relâchement de Q est obtenu en appliquant des généralisations g_k fondées sur $\mathcal{O}$: $g_k = a_k \rightsquigarrow b_k$ est une substitution telle que $(a_k \Rightarrow b_k) \in \mathcal{O}$. Par conséquent, $\Gamma(Q) = g_n(\ldots(g_1(Q))\ldots)$. Et donc, la remémoration donne un chemin de similarité :

$$idx(R) \models_{\mathcal{O}} \Gamma(Q) \xleftarrow{g_n} \ldots \xleftarrow{g_1} Q \tag{6}$$

Ce chemin de similarité est construit sur la base d'une recherche A* minimisant $\sum_k \texttt{coût}(g_k)$. Par exemple, la remémoration peut donner le résultat suivant :

$$\begin{aligned} Q &= \texttt{scarole} \wedge \texttt{jus\_de\_citron} \wedge \neg\texttt{oignon} \\ \Gamma(Q) &= \texttt{salade\_verte} \wedge \top \wedge \neg\texttt{oignon} \equiv \texttt{salade\_verte} \wedge \neg\texttt{oignon} \\ idx(R) &= \texttt{laitue} \wedge \texttt{vinaigre} \wedge \texttt{huile\_olive} \wedge \texttt{tomate} \wedge \textit{Rien d'autre} \end{aligned}$$

(Γ consiste à généraliser `scarole` en `salade_verte` et à supprimer `jus_de_citron` de la requête en le généralisant, en plusieurs étapes, en $\top$, la racine de la hiérarchie).

On peut noter que la remémoration donne un premier appariement : un chemin de similarité est une sorte de chemin d'adaptation n'impliquant aucune substitution $\sigma \in \texttt{CA}$ (que des substitutions issues de $\mathcal{O}$). Par conséquent, la recette remémorée R peut être adaptée en suivant ce chemin de similarité. Cependant, durant l'adaptation, certaines substitutions $\sigma \in \texttt{CA}$ peuvent être utilisées, et, quand c'est le cas, le résultat demandera moins d'effort d'adaptation et devrait être meilleur[14].

6 Choix des coûts

Les coûts associés aux substitutions sont utilisés pour la remémoration et pour la phase d'appariement de l'adaptation. Changer cette fonction de coût modifie donc le comportement du moteur de RÀPC. Par exemple, affecter un coût infini à une substitution la rend inapplicable. Quelle est la meilleur fonction de coût ? Cette question, à supposer qu'elle ait un sens[15] est difficile et nous ne prétendons pas que nous y répondrons. Il faut néanmoins faire un choix de cette fonction qui soit, autant se faire se peut, argumenté. Cette section a pour but de mettre en évidence quelques notions relatives à la fonction de coût, à son choix et à son implantation dans WIKITAAABLE.

6.1 Coût et effort d'adaptation

On trouve, par-ci par-là, dans la littérature consacrée au RÀPC, la notion d'« effort d'adaptation » (voir, par exemple [13]) sans que, à notre connaissance elle ait été définie quelque part de façon précise et générale. Nous allons néanmoins nous appuyer sur cette notion intuitive dans le cadre de l'application WIKITAAABLE pour lier les notions d'adaptation et de coût. Toujours dans un modèle binaire et supposé résulter d'un consensus, dans lequel les recettes sont soit bonnes soit mauvaises, on considérera que plus l'effort d'adaptation d'une bonne recette est grand, moins il est probable que la recette adaptée soit bonne[16]. On définit alors le coût d'un

[14] Si la fonction de coût est une estimation de l'effort d'adaptation, alors la recette adaptée devrait être meilleure selon (3) que selon (6). En effet, puisque rajouter les substitutions de `CA` ne permet que de créer de nouveaux chemins, on peut en déduire que $\sum_i \texttt{coût}(\sigma_i) + \sum_j \texttt{coût}(\gamma_j) \leqslant \sum_k \texttt{coût}(g_k)$.

[15] Ce qui se ramène à la question de la signification de ce qu'est une « bonne recette » : voir note 12.

[16] L'introduction de la notion de probabilité (à lier ou non à la théorie du même nom) est importante ici. Si on avait écrit « plus l'effort d'adaptation est grand, moins la recette adaptée est bonne », non seulement on quittait le modèle binaire bonne/mauvaise recette, ce qui serait un moindre mal, mais cette affirmation serait souvent mise en défaut. Par exemple, si une adaptation conduit de la recette R à la recette R'' en passant par la recette R', il se peut que R et R'' soient bonnes alors que R' ne l'est pas (exemple subjectif : R est une recette de tarte aux rhubarbes, $R' = (\texttt{rhubarbe} \rightsquigarrow \texttt{banane})(R)$ et $R'' = (\texttt{banane} \rightsquigarrow \texttt{pomme})(R')$).

chemin d'adaptation comme étant une mesure de l'effort du processus d'adaptation dont ce chemin est une représentation. Si l'adaptation est nulle (i.e., $\mathit{idx}(R) \models_{\mathcal{O}} Q$), alors ce coût est minimal et fixé à 0. Par ailleurs, on fait l'hypothèse que cette mesure est additive, autrement dit que le coût du chemin donné par (7) est la somme des coûts des $p + q$ chemins d'adaptation élémentaires. De plus, on fait l'hypothèse simplificatrice que chacun de ces coûts n'est fonction que de la substitution. Par conséquent

$$\texttt{coût}\left(R \xmapsto{\sigma_1} \ldots \xmapsto{\sigma_p} \Sigma(R) \xmapsto{\gamma_q^{-1}} \ldots \xmapsto{\gamma_1^{-1}} \Gamma^{-1} \circ \Sigma(R)\right) = \sum_{i=1}^{p} \texttt{coût}(\sigma_i) + \sum_{j=1}^{q} \texttt{coût}(\gamma_j^{-1}) \tag{7}$$

Enfin, on pose $\texttt{coût}(\gamma_j) = \texttt{coût}(\gamma_j^{-1})$ (ce qui n'est qu'un changement de notation destiné à ne pas trimballer « $^{-1}$ » partout) et on obtient la valeur à minimiser lors de l'appariement.

Cette hypothèse d'additivité s'écrit, plus généralement, pour deux substitutions σ_1 et σ_2 :

$$\texttt{coût}(\sigma_2 \circ \sigma_1) = \texttt{coût}(\sigma_1) + \texttt{coût}(\sigma_2) \tag{8}$$

Dans la suite de cette section est d'abord étudié le coût des substitutions par spécialisation selon l'ontologie puis celui des substitutions $\sigma \in \texttt{CA}$.

6.2 Coût des spécialisations γ^{-1}

Le coût d'une spécialisation γ^{-1} est noté $\texttt{coût}(\gamma)$, pour simplifier et pour faire le lien avec le coût des généralisations, tel qu'utilisé aux sections 4 et 5.

Soit $\gamma_1 = a \rightsquigarrow b$ et $\gamma_2 = b \rightsquigarrow c$ (la partie droite de la première substitution est la partie gauche de la deuxième). On suppose que γ_1 et γ_2 sont deux généralisations, autrement dit $a \models_{\mathcal{O}} b$ et $b \models_{\mathcal{O}} c$. On s'intéresse au coût $\texttt{coût}(\gamma_i)$ (et donc, à la mesure de l'effort d'adaptation γ_i^{-1}). La propriété d'additivité des coûts pour la composition des substitutions (8) et le fait que $\gamma_2 \circ \gamma_1 = a \rightsquigarrow c$ entraînent que

$$\texttt{coût}(a \rightsquigarrow c) = \texttt{coût}(a \rightsquigarrow b) + \texttt{coût}(b \rightsquigarrow c) \tag{9}$$

Il est donc nécessaire que la fonction de coût vérifie cette égalité.

Notons par ailleurs que cette égalité traduit une certaine robustesse de la fonction de coût vis-à-vis des ajouts de concepts dans l'ontologie. Par exemple, supposons que dans l'ontologie $\mathcal{O}$ on « intercale » la classe `agrume_jaune` entre la classe `citron` et la classe `agrume`. Autrement dit, on remplace l'axiome $\texttt{citron} \Rightarrow \texttt{agrume}$ par les deux axiomes $\texttt{citron} \Rightarrow \texttt{agrume\_jaune}$ et $\texttt{agrume\_jaune} \Rightarrow \texttt{citron}$. L'égalité (9) implique qu'il faut que

$$\texttt{coût}(\texttt{citron} \rightsquigarrow \texttt{agrume}) = \texttt{coût}(\texttt{citron} \rightsquigarrow \texttt{agrume\_jaune}) + \texttt{coût}(\texttt{agrume\_jaune} \rightsquigarrow \texttt{agrume})$$

Autrement dit, la généralisation qui était directe de `citron` à `agrume` a le même coût que la nouvelle généralisation de `citron` à `agrume` qui se fait via `agrume_jaune`.

Cette propriété désirée condamne le choix d'un coût constant (i.e., $\texttt{coût}(a \rightsquigarrow b) = C$ pour tout axiome $(a \Rightarrow b) \in \mathcal{O}$), ce qui reviendrait pour mesurer le coût de la généralisation $a \rightsquigarrow b$ à compter le nombre d'arcs de a vers b.

En revanche, la première définition du coût d'une généralisation $a \rightsquigarrow b$ ($a \models_{\mathcal{O}} b$) définie ci-dessous vérifie la propriété (9) :

$$\texttt{coût}(a \rightsquigarrow b) = K\,(\mu(b) - \mu(a))$$

$$\text{où } K > 0 \text{ est une constante, où } \mu(x) = \frac{\mathcal{N}(x)}{\mathcal{N}(\top)} \text{ pour toute classe } x \text{ de } \mathcal{O}$$

$$\text{et où } \mathcal{N}(x) = \text{le nombre de recettes indexées par } x = \text{card}\{R \text{ du livre de recettes} \mid \mathit{idx}(R) \models_{\mathcal{O}} x\}$$

Par exemple, s'il y a $\mathcal{N}(\top) = 1000$ recettes dans la base de cas dont $\mathcal{N}(\texttt{citron}) = 100$ recettes avec du citron comme ingrédient et $\mathcal{N}(\texttt{agrume}) = 300$ recettes avec un ou des agrumes comme ingrédient (que ce soit des citrons ou pas), alors, $\texttt{coût}(\texttt{citron} \rightsquigarrow \texttt{agrume}) = K.(300 - 100)/1000 = 0,2K$.

Au-delà de la satisfaction de la propriété (9) on peut comprendre cette définition comme suit. On cherche à définir μ, une mesure sur Ω, « l'espace de toutes les recettes ». La classe x correspond à un sous-ensemble $\mathcal{E}(x)$, p. ex., $\mathcal{E}(\texttt{citron}) \subseteq \Omega$ est l'ensemble de toutes les recettes avec du citron et $\mathcal{E}(\top) = \Omega$. On approche Ω par l'ensemble fini des recettes de la base de cas, et on propose de donner une mesure $\mu(x)$ proportionnelle à $\mathcal{N}(x)$ et normalisée ($\mu(\top) = 1$). La différence $\mu(b) - \mu(a)$ correspond à la mesure de l'ensemble des recettes indexées par b mais pas par a, i.e., à la mesure de $\mathcal{E}(b) \setminus \mathcal{E}(a)$. Ainsi, le coût de la spécialisation $\gamma^{-1} = \texttt{agrume} \rightsquigarrow \texttt{citron}$ est le produit de la constante $K/\mathcal{N}(\top)$ et du nombre de recettes contenant des agrumes mais pas de citrons : si le chemin d'adaptation de coût minimal est $idx(R) \vDash_{\mathcal{O}} \Gamma(Q) \overset{\texttt{agrume}\rightsquigarrow\texttt{citron}}{\longmapsto} Q$, cela signifie que R est une recette satisfaisant tous les critères de Q, à part « citron » mais satisfaisant « agrume » à la place. L'effort de cette adaptation est alors proportionnel à la mesure de la différence $\mathcal{E}(\texttt{agrume}) \setminus \mathcal{E}(\texttt{citron})$.

Cette première définition de la fonction de coût a été améliorée. En effet, une spécialisation de recette $a \rightsquigarrow b$ (avec $b \vDash_{\mathcal{O}} a$) se fait de façon très différente selon que a et b sont des propriétés liées à des ingrédients (p. ex., $a = \texttt{agrume}$, $b = \texttt{citron}$) ou pas (p. ex., $a = \texttt{Asie}$, $b = \texttt{Chine}$). Dans le premier cas, il y a effectivement modification de la recette : on substitue un agrume qui n'est pas un citron (p. ex., une orange) par un citron. L'adaptation conduit à une correction de la recette pour qu'elle satisfasse la requête et le coût est relatif à l'erreur de correction. Dans le deuxième cas, en fait, on ne modifie pas la recette asiatique et non chinoise (p. ex. vietnamienne ou indienne) en une recette chinoise : on propose la première comme une recette (très) approximativement chinoise. La probabilité de donner un mauvais résultat nous semble beaucoup plus importante dans le deuxième cas que dans le premier[17]. C'est pourquoi on va faire dépendre le coefficient K des *points de vue* de a et b. Si a et b sont « du point de vue ingrédient » ($\texttt{pdv}(a) = \texttt{pdv}(b) = \texttt{pdv\_ingrédient}$, i.e., ils représentent des classes de recettes définies par un type d'ingrédient) K sera plus faible que si a et b sont « du point de vue géographique » ($\texttt{pdv}(a) = \texttt{pdv}(b) = \texttt{pdv\_géographique}$). Le cas où a et b sont de points de vue différents conduit à interdire la spécialisation $a \vDash_{\mathcal{O}} b$ (coût infini). Le travail théorique sur la représentation « propre » des points de vue est en cours et s'inspire de TROEPS [11] et de la partie du travail de thèse de Mathieu d'Aquin sur les points de vue [9, 10]. Cette réflexion a conduit à la définition actuelle de la fonction de coût :

$$\texttt{coût}(a \rightsquigarrow b) = \begin{cases} K_{\texttt{pdv}(a)}\,(\mu(b) - \mu(a)) & \text{si } \texttt{pdv}(a) = \texttt{pdv}(b) \\ +\infty & \text{sinon} \end{cases}$$

où $K_{\texttt{pdv}} > 0$ ne dépend que du point de vue pdv

6.3 Coût des $\sigma \in \texttt{CA}$

Une réflexion de fond reste à mener pour l'estimation de $\texttt{coût}(\sigma)$ pour $\sigma \in \texttt{CA}$. Il nous semble simplement que ces coûts doivent être en général inférieurs (voire très inférieurs) à ceux des coûts des spécialisations γ^{-1} évoqués ci-dessus. En effet, les $\sigma \in \texttt{CA}$ sont issus de connaissances expertes en lien avec les « substitutions d'ingrédients culinairement autorisés » alors que les γ^{-1} sont liées à la fois à l'organisation de l'ontologie et (dans certains cas, assez rares) à de telles substitutions culinairement autorisées (p. ex., la classe beurre_ou_margarine et les axiomes $\texttt{beurre} \Rightarrow \texttt{beurre\_ou\_margarine}$ et $\texttt{margarine} \Rightarrow \texttt{beurre\_ou\_margarine}$ ont été introduits pour faciliter les substitutions $\texttt{beurre} \rightsquigarrow \texttt{margarine}$ et $\texttt{margarine} \rightsquigarrow \texttt{beurre}$).

[17] On peut, pour comprendre intuitivement cette idée, passer par la métaphore des fonctions régulières $f : \mathbb{R} \longrightarrow \mathbb{R}$ par des développements de Taylor (métaphore utilisée également dans [15]). Une approximation de $f(x_1)$ à partir de $f(x_0)$ selon un développement du premier ordre donne $f(x_1) \simeq f(x_0) + (x_1 - x_0)f'(x_0)$ avec une erreur majorée par $M_2(x_1 - x_0)^2$ (où $M_2 = \max\{f''(x) \mid x \in [x_0; x_1]\}$). À l'ordre 0, on obtient : $f(x_1) \simeq f(x_0)$ avec une erreur majorée par $M_1|x_1 - x_0|$ (où $M_1 = \max\{f'(x) \mid x \in [x_0; x_1]\}$). Dans le premier cas, il y a une modification de $f(x_0)$ pour obtenir une valeur approchée de x_1, dans le deuxième, $f(x_0)$ est utilisée comme valeur approchée de $f(x_1)$. La majoration de l'erreur (pour $|x_1 - x_0|$ suffisamment petit) sera plus petite (et donc meilleure), dans le premier cas que dans le deuxième. Dans cette métaphore, (x_0, x_1) correspond à (a, b), i.e., à (agrume, citron) dans le premier cas et à (Asie, Chine) dans le deuxième.

6.4 Comment atténuer l'effet de l'arbitraire dans le choix des coûts

Même si nous avons donné une tentative de justification dans le choix des coûts, nous n'avons certainement pas la fonction de coût idéale : nous avons fait des hypothèses simplificatrices et nous avons laissé le choix des coefficients $K_{\texttt{pdv}(a)}$ ouvert. Par ailleurs, la remémoration n'utilise pas les $\sigma \in \texttt{CA}$: il se peut qu'une recette qui, à une substitution $\sigma \in \texttt{CA}$ près, réponde à la requête nécessite beaucoup de généralisations g_k pour être mise en évidence. Par conséquent, la meilleure remémoration de recette dans l'absolu (si, encore une fois, on suppose que cet absolu existe) ne sera pas nécessairement effectué : l'algorithme de remémoration cherche les recettes demandant une modification de la requête qui soit minimale au sens de l'imparfaite fonction de coût.

Pour augmenter les chances de trouver la meilleure recette, une modification de l'algorithme de remémoration a été implantée. Elle consiste simplement à poursuivre la remémoration après la mise en évidence de la recette la plus proche au sens de la fonction de coût : on ne cherche plus seulement les recettes R qui répondent à une modification de coût minimal $\mathcal{C}_*$ mais celles qui répondent à toute modification $\Gamma(Q)$ de la requête telle que $\texttt{coût}(\Gamma) \in [\mathcal{C}_*; \mathcal{C}_* + J]$ où $J \geqslant 0$ est un paramètre de la remémoration[18].

7 Vers un nouveau moteur

La description ci-dessus du moteur de RÀPC de WIKITAAABLE se veut un moyen de poser les bases de la prochaine version du système. Cette section liste quelques modifications prévues, les problèmes de représentation qu'elles posent et comment il est envisagé de résoudre ces problèmes.

Des cas-recettes spécifiques aux cas-recettes généraux. WIKITAAABLE considère actuellement qu'une recette correspond à la description d'une expérience particulière (une instance). Ainsi, une recette $R_{\texttt{tap}}$ de tarte aux pommes est interprétée comme la description d'un processus ayant eu lieu et s'appuyant sur des pommes déterminées (les 4 premières pommes du $6^{\text{ème}}$ pommier de Marcel, en 2008). Une interprétation différente consiste à considérer cette recette comme un algorithme avec des paramètres (4 pommes, de n'importe quels types[19]) qui peut être exécuté autant de fois que nécessaire. L'avantage de cette nouvelle interprétation est qu'elle permet de prendre en compte la généralité des recettes. La recette $R_{\texttt{tap}}$ sera spécialisable en une recette $R_{\texttt{tacp}}$ de tarte aux « courts pendus » (variété de pommes cueillie sur le Web ; $(\texttt{court\_pendu} \Rightarrow \texttt{pomme}) \in \mathcal{O}$). Cela a des conséquences sur les inférences et sur la représentation.

Conséquences sur les inférences. Soit la requête $Q = \texttt{tarte} \wedge \texttt{court\_pendu}$. Avec le moteur actuel de WIKITAAABLE, la recette $R_{\texttt{tap}}$ supposera une généralisation de la requête lors de la remémoration. En effet, $\mathit{idx}(R_{\texttt{tap}}) = \texttt{tarte} \wedge \texttt{pomme} \not\models_{\mathcal{O}} Q$. Un chemin de similarité sera $\mathit{idx}(R_{\texttt{tap}}) \models_{\mathcal{O}} g(Q) \longmapsto Q$ où $g = \texttt{court\_pendu} \rightsquigarrow \texttt{pomme}$. L'adaptation selon ce chemin de similarité donnera la recette $R_{\texttt{tacp}}$ obtenue en substituant `pomme` par `court_pendu`. Si ce résultat semble intuitivement correct, on peut contester le fait qu'il demande une adaptation : ce n'est qu'une application de $R_{\texttt{tap}}$ à un type de pomme particulier.

Pour une prochaine version de WIKITAAABLE, voici comment nous envisageons cette inférence. Au lieu de faire le test « $\Sigma(\mathit{idx}(R)) \models_{\mathcal{O}} \Gamma(Q)$ », on fera le test « $\Sigma(\mathit{idx}(R)) \wedge \Gamma(Q)$ est satisfiable (étant donné $\mathcal{O}$) ». Ainsi, dans l'exemple précédent

$\mathit{idx}(R_{\texttt{tap}}) \wedge Q = \texttt{tarte} \wedge \texttt{pomme} \wedge \texttt{tarte} \wedge \texttt{court\_pendu} \equiv_{\mathcal{O}} \texttt{tarte} \wedge \texttt{court\_pendu}$ qui est satisfiable

La remémoration donnera alors la recette $R_{\texttt{tap}}$ qui répond à la requête en substituant `pomme` par la variété `court_pendu` ou, pour le dire autrement, sans raisonnement par analogie mais à l'aide d'une déduction.

Cette nouvelle inférence permet aussi de tenir compte de recettes avec des alternatives (p. ex., l'ingrédient « beurre ou margarine »).

[18] Selon un des relecteurs de cet article, « L'utilisation de la notion de coût limite le risque d'adaptation mais nuit à la créativité. » Si on adhère à ce point de vue, l'imperfection du choix des coûts pourrait être considéré positivement comme source de créativité. Encore faut-il adhérer à ce point de vue, qui est en débat depuis longtemps dans la communauté RÀPC.

[19] Avec comme sous-entendu une normalisation selon leur taille : 4 pommes standard correspondent à 6 pommes rainettes.

Conséquences sur la représentation. La façon dont l'hypothèse du monde clos est appliquée sur les index de recettes entraîne qu'une seule interprétation est possible. Dans la prochaine version du système, cela ne doit plus être vrai puisque une recette R représentera un ensemble d'instances de recettes (p. ex., $R_{\texttt{tap}}$ a pour instance l'interprétation $\mathcal{I}$ de tarte aux court-pendus sans rainette — $\texttt{court\_pendu}^{\mathcal{I}} = \texttt{vrai}$, $\texttt{rainette}^{\mathcal{I}} = \texttt{faux}$ — et l'interprétation $\mathcal{J}$ de tarte aux rainettes sans court-pendu — $\texttt{court\_pendu}^{\mathcal{J}} = \texttt{faux}$, $\texttt{rainette}^{\mathcal{J}} = \texttt{vrai}$). Mais abandonner complètement l'hypothèse du monde clos n'est pas acceptable. On veut toujours pouvoir inférer que $\mathit{idx}(R_{\texttt{tap}}) \models_{\mathcal{O}} \neg\texttt{viande} \wedge \neg\texttt{poisson}$ (ce n'est parce qu'on est végétarien qu'on ne mange pas de tartes aux pommes). Par conséquent, il faudra trouver un moyen pour exprimer l'hypothèse du monde clos différemment...

La solution actuellement envisagée est la suivante. On définit comme avant $\mathit{idx}(R)$ par une conjonction $C = a_1 \wedge a_2 \wedge \ldots \wedge a_m$ de littéraux positifs d'où on infère $R = \neg b_1 \wedge \neg b_2 \wedge \ldots \wedge \neg b_n$ une conjonction de littéraux négatifs. $\mathit{idx}(R)$ sera alors équivalent à $C \wedge R$ (R est le « Rien d'autre » de l'équation (1)). Dans la version actuelle, les b_i sont les classes b telles que $\mathit{idx}(R) \not\models_{\mathcal{O}} b$ (par exemple, $b = \texttt{viande}$, mais aussi $b = \texttt{court\_pendu}$). Dans la nouvelle version, les b_i devraient être les b tels que de plus, pour tout i, $b \not\models_{\mathcal{O}} a_i$. Cette nouvelle version est encore peu justifiée théoriquement (elle résulte d'un bricolage) et reste à étudier...

Extension du langage à l'aide d'attributs. Actuellement, ni les quantités des ingrédients dans les recettes (masses, nombres d'unités) ni les propriétés numériques des ingrédients (nombre de calories d'une myrtille, « pouvoir sucrant » de 1 g de miel) ne sont représentables au sein du moteur d'inférences. Pourtant, ce genre d'informations peut être utile, par exemple pour répondre à une requête demandant une recette faiblement calorique (la solution actuelle est peut satisfaisante : elle consiste à éviter certains ingrédients, indépendamment de leurs quantités), pour substituter 4 poires par 3 pommes et 10 g de sucres (pour conserver la masse et le pouvoir sucrant), etc.[20]

Par ailleurs, actuellement seules des classes de recettes sont représentées : les classes `ananas` et `dessert` représentent respectivement les recettes avec de l'ananas et les recettes de desserts. Ainsi, on a l'axiome quelque peu surprenant $\texttt{jus\_de\_citron} \Rightarrow \texttt{citron}$ ce qui s'interprète par « Toute recette avec du jus de citron est une recette avec du citron. » et non par « Une instance de jus de citron est une instance de citron. »

Ces deux considérations nous incitent à passer à un formalisme utilisant des attributs[21]. Une logique de descriptions peu expressive (et dont les inférences sont polynomiales) telle que $\mathcal{EL}(\mathtt{D})$, avec un domaine concret permettant d'exprimer des contraintes numériques, semble adaptée à nos besoin. La difficulté principale sera de gérer dans ce formalisme d'une part l'hypothèse du monde clos et d'autre part la non-monotonie désirée de certaines inférences, telle que décrite ci-dessous.

Gérer la non-monotonie et la typicalité. Le lait est-il un liquide ? Le beurre est-il gras ?[22] Pas nécessairement : il y a du lait en poudre ($(\texttt{lait\_en\_poudre} \Rightarrow \texttt{lait}) \in \mathcal{O}$) et du beurre allégé ($(\texttt{beurre\_allégé} \Rightarrow \texttt{beurre}) \in \mathcal{O}$). Par conséquent, $\mathcal{O}$ doit respecter le fait que $\texttt{lait} \not\models_{\mathcal{O}} \texttt{liquide}$ et $\texttt{beurre} \not\models_{\mathcal{O}} \texttt{aliment\_gras}$. Oui, mais, *typiquement*, le lait est liquide et le beurre est gras. Donc, *sauf information contraire*, une recette ayant du lait comme ingrédient aura un ingrédient liquide, une recette ayant du beurre comme ingrédient aura un ingrédient gras.

Nous envisageons de représenter cela de la façon suivante. À chaque classe (p. ex., `lait`) on associe une instance de cette classe qui va jouer le rôle de prototype (p. ex., `lait_prototypique`) et auquel on associe les propriétés typiques du lait (par exemple, le fait que `lait_prototypique` soit également une instance de `liquide`)[23]. Comment cela s'articule avec les inférences du système est un (ensemble de) problème(s) ouvert(s)...

[20] Ce genre d'adaptation est à l'étude actuellement [6].

[21] Ou des rôles, ou des propriétés : ce sont pratiquement des synonymes, attribut est plus typique des langages de représentation par objets, rôle, des logiques de descriptions et propriété, des standards du Web sémantique comme RDF(S) ou OWL.

[22] Autres questions du même genre : « Le pastis est-il alcoolisé ? », « Le café est-il un excitant ? », « Les oiseaux volent-ils ? »

[23] Cette idée vient de discussions avec Hala Skaf-Molli et Pascal Molli à propos de la représentation des points de vue compte tenu des restrictions du langage utilisé (pour être précis, ces restrictions viennent du fait que l'export doit respecter les contraintes de OWL DL). Il n'est en effet pas possible d'associer directement à une classe (telle que `citron`) une propriété (telle que le point de vue ingrédient). L'astuce consiste alors à associer ce point de vue à l'instance `citron_prototypique` de `citron`.

Représentation des préparations. Ce qui précède concerne uniquement la représentation des ingrédients des recettes. Il est envisagé de représenter aussi les préparations. Une piste dans cette voie est décrite dans [12] et consiste à représenter les préparations par des étapes de préparations et des contraintes temporelles entre ces étapes.

8 Conclusion

Le moteur de RàPC du système WIKITAAABLE a été initialement conçu selon les principes d'autres moteurs de classification et de RàPC (notamment ceux de RÉSYN/RàPC [14] et des premières versions du système KASIMIR [5], dans les domaines d'application respectifs de la chimie organique et de la cancérologie). Puis, il s'en est différencié pour s'adapter aux contraintes de l'application, par exemple par l'introduction de l'hypothèse du monde clos pour les classes représentant des index de recettes, hypothèse rendue nécessaire par la possibilité d'avoir des littéraux négatifs dans les requêtes : sans cette hypothèse, rien n'indique que la recette du gaspacho ne contient pas de viande (exemple repris de [9]). À présent que les deux premières versions du moteur de RàPC pour le projet TAAABLE ont été développées, et avant d'en développer une troisième, il a semblé nécessaire de faire une description détaillée de l'état actuel du moteur. Cette description soulève des problèmes scientifiques, tels que l'estimation des coûts, qui sont abordés dans cet article avec la solution actuellement implantée.

La perspective principale de ce travail a été développée à la section 7 : comment passer à la version suivante du moteur, qui devrait intégrer une extension de l'expressivité du langage de représentation, tout en restant efficace en termes de temps de calcul. Un travail qui est en court depuis le début du projet TAAABLE est celui de l'acquisition des connaissances pour ce système. Ces connaissances doivent être appréhendées du point de vue du système qui les manipule, i.e., du point de vue du raisonnement artificiel. Le développement *conjoint* du moteur et de méthodes et outils pour l'acquisition et extraction de connaissances doit se poursuivre tout au long du projet TAAABLE (i.e., jusqu'en 2048).

Remerciements

L'auteur tient à remercier les personnes impliquées dans le projet TAAABLE pour le plaisir qu'il a à travailler avec elles. Il remercie également les trois relecteurs pour leurs suggestions.

Références

[1] Badra (F.), Bendaoud (R.), Bentebitel (R.), Champin (P.-A.), Cojan (J.), Cordier (A.), Desprès (S.), Jean-Daubias (S.), Lieber (J.), Meilender (T.), Mille (A.), Nauer (E.), Napoli (A.) et Toussaint (Y.). – Taaable : Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. *In : ECCBR Workshops*, pp. 219–228. – 2008.

[2] Badra (F.), Cordier (A.) et Lieber (J.). – Découverte opportuniste de connaissances d'adaptation. *In : À paraître dans les actes du 17ème atelier raisonnement à partir de cas (RàPC-09)*. – 2009.

[3] Badra (F.), Cordier (A.) et Lieber (J.). – Opportunistic Adaptation Knowledge Discovery. *In : Case-Based Reasoning Research and Development / ICCBR 2009*. – 2009. To appear.

[4] Bergmann (R.) et Wilke (W.). – Building and Refining Abstract Planning Cases by Change of Representation Language. *Journal of Artificial Intelligence Research*, vol. 3, 1995, pp. 53–118.

[5] Bresson (B.) et Lieber (J.). – Classification pour l'aide au traitement du cancer du sein. *In : Septième journées de la Société Francophone de Classification, SFC'99*, éd. par Le Ber (F.), Mari (J.-F.), Napoli (A.) et Simon (A.). pp. 53–59. – Nancy, septembre 1999.

[6] Cojan (J.) et Lieber. (J.). – Belief Merging-based Case Combination. *In : Case-Based Reasoning Research and Development / ICCBR 2009*. – 2009. To appear.

[7] Cordier (A.). – *Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning*. – France, Thèse de PhD, Université Lyon 1, november 2008.

[8] Cordier (A.), Lieber (J.), Molli (P.), Nauer (E.), Skaf-Molli (H.) et Toussaint (Y.). – Wikitaaable : A semantic wiki as a blackboard for a textual case-based reasoning system. *In : SemWiki 2009 – 4th Semantic Wiki Workshop*. – Heraklion, Greece, May 2009.

[9] d'Aquin (M.). – *Un portail sémantique pour la gestion des connaissances en cancérologie*. – Thèse d'université, Université Henri Poincaré Nancy 1, soutenue le 15 décembre, 2005.

[10] d'Aquin (M.), Lieber (J.) et Napoli (A.). – Decentralized Case-Based Reasoning for the Semantic Web. *In : Proceedings of the 4th International Semantic Web Conference (ISWC 2005)*, éd. par Gil (Yolanda) et Motta (Enrico). pp. 142–155. – Springer, November 2005.

[11] Euzenat (J.). – Représentation de connaissances par objets. *In : Langages et modèles à objets — État des recherches et perspectives*, éd. par Ducournau (R.), Euzenat (J.), Masini (G.) et Napoli (A.), pp. 293–319. – Le Chesnay, INRIA, 1998.

[12] Le Ber (F.), Lieber (J.) et Napoli (A.). – Utilisation d'une algèbre temporelle pour la représentation et l'adaptation de recettes de cuisine. *In : À paraître dans les actes du 17ème atelier raisonnement à partir de cas (RàPC-09)*. – 2009.

[13] Leake (D. B.), Kingley (A.) et Wilson (D.). – Linking Adaptation and Similarity Learning. *In : Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*. – Lawrence Erlbaum, Inc., 1996.

[14] Lieber (J.). – *Raisonnement à partir de cas et classification hiérarchique. Application à la planification de synthèse en chimie organique*. – Thèse d'université, Université Henri Poincaré Nancy 1, 1997.

[15] Lieber (J.). – La méthode d'Euler vue comme une application du raisonnement à partir de cas. *In : Actes du* VII*ème séminaire français de raisonnement à partir de cas*, éd. par Mille (A.) et Trousse (B.). – 1999.

[16] Lieber (J.). – Strong, Fuzzy and Smooth Hierarchical Classification for Case-Based Problem Solving. *In : Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-02), Lyon, France*, éd. par van Harmelen (F.). pp. 81–85. – IOS Press, Amsterdam, 2002.

[17] Lieber (J.) et Napoli (A.). – Vers une proposition de définitions des notions d'abstraction et de généralisation dans le cadre du raisonnement à partir de cas appliqué à la résolution de problèmes. *In : Actes du* VI*ème séminaire français de raisonnement à partir de cas*, éd. par Malek (M.). – 1998.

[18] Melis (E.), Lieber (J.) et Napoli (A.). – Reformulation in Case-Based Reasoning. *In : Fourth European Workshop on Case-Based Reasoning, EWCBR-98*, éd. par Smyth (B.) et Cunningham (P.). pp. 172–183. – Springer, 1998.

[19] Richter (M. M.). – Introduction. *In : Case-Based Reasoning Technologies. From Foundations to Applications*, éd. par Lenz (M.), Bartsch-Spörl (B.), Burkhard (H.-D.) et Wess (S.), chap. 1, pp. 1–15. – Springer, 1998.

[20] Riesbeck (C. K.) et Schank (R. C.). – *Inside Case-Based Reasoning*. – Hillsdale, New Jersey, Lawrence Erlbaum Associates, Inc., 1989.

[21] Smyth (B.) et Keane (M. T.). – Using adaptation knowledge to retrieve and adapt design cases. *Knowledge-Based Systems*, vol. 9, n2, 1996, pp. 127–135.

Dissimilarité analogique et apprentissage d'arbres

Anouar Ben Hassena, Laurent Miclet
IRISA / Université de Rennes 1 - ENSSAT
6 rue de Kerampont, B.P. 80518, F-22305 Lannion Cedex
{anouar.ben-hassena, laurent.miclet}@enssat.fr

Résumé

Nous présentons dans cet article une nouvelle définition d'une mesure de dissimilarité entre structures arborescentes : la dissemblance analogique. Celle-ci est fondée sur un alignement analogique de quatre structures d'arbres. Nous présentons la définition de la proportion analogique entre arbres, son extension à la dissimilarité analogique et nous proposons sa mise en œuvre par un algorithme polynomial basé sur un alignement multi-arbres. Nous montrons ensuite comment utiliser cette notion et cet algorithme dans une application d'apprentissage.

1 Introduction

La notion d'analogie, en particulier sous la forme de la proportion analogique, a été étudiée comme une des modalités du raisonnement depuis Aristote [10, 8]. Il s'agit également d'un raisonnement par généralisation, ni abductif ni inductif, qui modélise une troisième forme d'apprentissage. Dans le cas de l'apprentissage supervisé, de manière élémentaire, il s'appuie sur un raisonnement du type suivant. Je sais que les objets a et b appartiennent à la classe ω_1 et l'objet c à la classe ω_2. J'observe que je peux transformer a en b (ou apparier a avec b) de la même manière que c en x. Par conséquent, je propose d'attribuer la classe ω_2 à x.

Son application en intelligence artificielle a été envisagée et testée très tôt, entre autres dans [7], mais ce n'est que plus récemment que le raisonnement à partir de cas a développé des réalisations opérationnelles de certaines formes particulières du raisonnement analogique (voir par exemple [1]). Récemment, un intérêt croissant s'est manifesté pour un point de vue plus formel sur l'analogie, la *proportion analogique*. Cette notion est en effet définie rigoureusement et son application dans des espaces de représentation de natures variées a été développée avec des résultats opérationnels intéressants. Son application à l'apprentissage et à la génération est conceptuellement simple ; en revanche, comme dans bien des domaines de l'intelligence artificielle, certains problèmes de complexité algorithmique restent à surmonter.

Nous nous intéressons dans cet article à des objets représentés par des arbres étiquetés et ordonnés, en particulier dans le cadre de la génération par apprentissage d'arbres prosodiques pour la synthèse de la parole [3]. En ce qui concerne les données structurées, la proportion analogique a déjà été appliquée aux séquences et aux arbres : les références [14] et [15, 12] en donnent les principes, des algorithmes et des applications. Le travail que nous présentons ici ne comporte pas de résultats expérimentaux : il vise à fonder une notion d'analogie entre arbres, un peu différente de celle qui vient d'être citée, afin de pouvoir l'appliquer ultérieurement à l'apprentissage et à la génération, comme cela a été fait pour les séquences en particulier dans [2, 12].

Le paragraphe 2 présente les mesures de similarité entre arbres fondées sur des opérations d'édition et propose une nouvelle formulation pour un algorithme d'alignement, qui permet d'étendre cette notion à celle de l'alignement de plus de deux arbres. Ensuite, dans le paragraphe 3, après avoir rappelé comment ces notions s'appliquent en pratique à l'apprentissage, nous définissons une proportion analogique et une dissimilarité entre arbres. Cette dernière notion permet de mesurer de combien quatre arbres « ratent » la proportion analogique. En conclusion, nous présentons nos travaux à venir dans le cadre de l'apprentissage par analogie dans les structures d'arbres.

2 Mesure de similarité entre arbres

2.1 Distance d'édition

Les objets sur lesquels nous voulons réaliser une tâche d'apprentissage sont des arbres ordonnés[1] et étiquetés. On peut donc identifier leurs nœuds par leur *position*, où la position est une séquence d'entiers (un mot dans $\mathbb{N}^+$), comme dans [4]. Nous nous intéressons d'abord aux mesures de similarité et aux distances appropriées à ces structures.
Les distances considérées dans nos travaux reposent sur le principe de l'*édition* d'un arbre en un autre, par la composition d'opérations élémentaires, en visant à minimiser leur coût cumulé. Le principe est similaire à celui de la distance entre séquences, comme dans [18]. On définit trois opérations élémentaires selon [16] comme $(l_1 \rightarrow l_2)$, où $(l_1,l_2) \in (\Sigma_\lambda \times \Sigma_\lambda) \setminus (\lambda,\lambda)$ avec Σ alphabet fini, λ le nœud vide et $\Sigma_\lambda = \Sigma \cup \{\lambda\}$. Ces opérations s'appellent *Insertion* si $l_1 = \lambda$, *Suppression* si $l_2 = \lambda$ et *Substitution* si $l_1 \neq \lambda$ et $l_2 \neq \lambda$. Ces opérations portent sur les étiquettes, mais nous pouvons aussi la faire porter sur les nœuds en définissant $(v \rightarrow w)$
par (étiquette(v) $\rightarrow$ étiquette(w)).

Ensuite, une fonction de coût c est définie, assignant à chaque opération d'édition $(l_1 \rightarrow l_2)$ une valeur réelle positive $c(l_1 \rightarrow l_2)$. Cette dernière est définie en fonction des étiquettes des nœuds mis en jeu lors des opérations correspondantes.
Ces opérations d'édition définissent alors un *mapping* comme un ensemble M de couples de nœuds (i, j), respectivement des arbres T_1 et T_2, et satisfaisant les conditions suivantes,
Pour toutes les paires (i_1,j_1) et (i_2,j_2) dans M :

1. $i_1 = i_2 \Longleftrightarrow j_1 = j_2$ (*one-to-one condition*).
2. i_1 est à gauche de [2] $i_2 \Longleftrightarrow j_1$ est est à gauche de j_2 (*sibling condition*).
3. i_1 est un ancêtre de $i_2 \Longleftrightarrow j_1$ est un ancêtre de j_2 (*ancestor condition*).

Le coût de mapping peut être défini ensuite par :

$$c(M) = \sum_{(i,j)\in M} c(i,j) + \sum_{i \notin M} c(i,\lambda) + \sum_{j \notin M} c(\lambda,j)$$

et la distance d'édition $dist$ entre T_1 et T_2 est alors définie par le coût minimal $c(M)$ telle que

$$dist(T_1,T_2) = min\ \{c(M) \mid M\ mapping\ de\ T_1\ vers\ T_2\}$$

L'étape suivante consiste à trouver ce mapping de coût minimal. Plusieurs algorithmes existent pour traiter cela. Historiquement, le premier algorithme pour la distance d'édition entre arbres a été proposé par [16], avec une complexité $O(|T_1| \times |T_2| \times L_1^2 \times L_2^2)$ en temps[3]. Il a été remplacé avantageusement par des algorithmes épurés, plus rapides et fondés également sur le principe de la programmation dynamique, par exemple [20, 9, 6, 5]. Nous donnons ici une récursion simple qui forme la base de ces algorithmes .

L'algorithme de base [16] La récursion suivante montre une manière de calculer la distance d'édition dans le cas général des forêts, où une forêt F est un ensemble fini ordonné d'arbres. Nous notons :

- $F[v]$ la forêt obtenue à partir de l'arbre de racine v en supprimant la racine v,
- $F - v$ la forêt obtenue par la suppression d'un nœud quelconque v de F,
- $F - T[v]$ la forêt obtenue par la suppression de v et de tous les descendants de v,
- θ la forêt vide.

[1] Un arbre est *ordonné* s'il existe un ordre parmi les fils de chaque nœud.

[2] Un nœud N de position $Np_1.....Np_n$ est *à gauche* d'un nœud M de position $Mp_1.....Mp_m$ ssi N est ni un ancêtre ni un descendant de M et $\exists\ i \in [1, min(n,m)]$ tel que $Np_1.....Np_{i-1} = Mp_1.....Mp_{i-1}$ et $Np_i < Mp_i$.

[3] $|T_1|$ est le nombre de nœuds du premier arbre et L_1 sa profondeur.

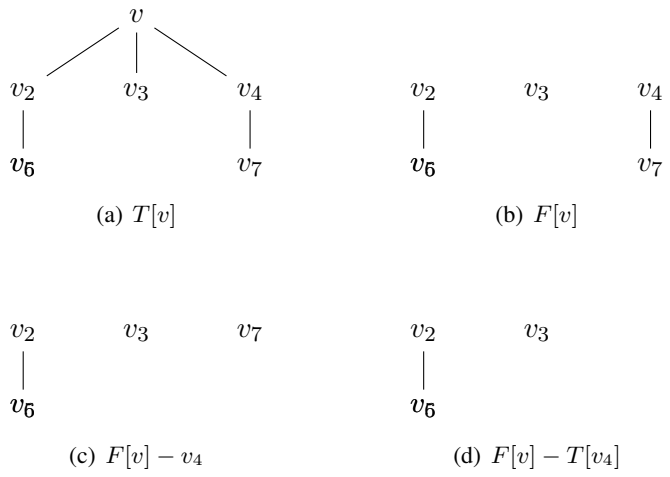


FIG. 1 – Terminologie pour les forêts

Récursion Soit F_1 et F_2 deux forêts ordonnées et c la fonction coût définie sur les nœuds. Soit v et w les racines des arbres les plus à droite (*rightmost*) dans F_1 et F_2 respectivement. On a,

$$
\begin{aligned}
&c(\theta,\theta) = 0\\
&c(F_1,\theta) = c(F_1 - v,\theta) + c(v \to \lambda)\\
&c(\theta,F_2) = c(\theta,F_2 - w) + c(\lambda \to w)\\
&c(F_1,F_2) = min \begin{cases} c(F_1 - v,F_2) + c(v \to \lambda)\\ c(F_1,F_2 - w) + c(\lambda \to w)\\ c(F_1[v],F_2[w]) + c(F_1 - T_1[v],F_2 - T_2[w]) + c(v \to w) \end{cases}
\end{aligned}
$$

Certains auteurs, comme [19, 13, 11], ont étudié des versions restreintes du problème et ont introduit la notion de distance d'édition sous contraintes sur les mappings possibles des sous arbres. L'idée est que la restriction sur le mapping engendre moins de sous problèmes et par conséquent conduit à un algorithme de programmation dynamique plus rapide.

2.2 Alignement

Une autre manière de calculer une ressemblance entre arbres ordonnés, par un mapping sous contraintes particulières, est *l'alignement*. Un alignement A de T_1, T_2 est un mapping qui est obtenu d'abord par l'insertion des nœuds d'étiquette λ dans T_1 et dans T_2 de telle façon qu'ils aient la même structure. Ensuite, les opérations d'éditions nécessaires pour passer d'un arbre à l'autre sont comptabilisées. Le coût d'un alignement A est la somme de tous les coûts des opérations dans A. La figure 2 montre un exemple d'alignement entre deux arbres ordonnés.

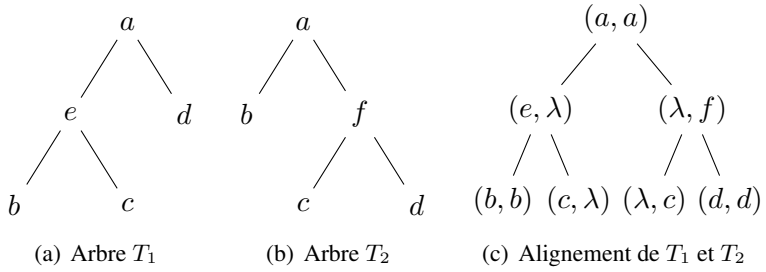


FIG. 2 – Exemple d'un Alignement A

L'alignement des arbres ordonnés a été introduit par [17]. Cet algorithme présente une complexité de $O(|T_1||T_2|(\text{ degré }(T_1)+\text{ degré}(T_2))^2)$ en temps et $O(|T_1||T_2|(\text{ degré }(T_1)+\text{ degré }(T_2)))$ en espace. Nous le détaillons et nous montrons comment calculer un alignement entre deux arbres.

L'algorithme de Jiang Soit $\alpha(T_1, T_2)$ le coût d'alignement des arbres T_1, T_2 et $\alpha(F_1, F_2)$ le coût d'alignement des forêts F_1, F_2.

Pour tout v un nœud de T_1 et w un nœud de T_2 avec $v_1, ..., v_i$ et $w_1, ..., w_j$ leurs fils respectifs. Nous avons :

$$\alpha(T_1(v), T_2(w)) = min \begin{cases} \alpha(T_1(v), \theta) + \min\limits_{1\leq r\leq i}\{\alpha(T_1(v_r), T_2(w)) - \alpha(T_1(v_r), \theta)\} \\ \alpha(\theta, T_2(w)) + \min\limits_{1\leq r\leq j}\{\alpha(T_1(v), T_2(w_r)) - \alpha(\theta, T_2(w_r))\} \\ \alpha(F_1(v), F_2(w)) + c(v, w) \end{cases}$$

Il reste maintenant à définir un alignement des forêts F_1 et F_2. Pour tout $1 \leq s \leq i$ et $1 \leq t \leq j$, nous avons,

$$\alpha(F_1(v_1, v_s), F_2(w_1, w_t)) =$$

$$\min \begin{cases} \alpha(F_1(v_1, v_{s-1}), F_2(w_1, w_t)) + \alpha(T_1(v_s), \theta) \\ \alpha(F_1(v_1, v_s), F_2(w_1, w_{t-1})) + \alpha(\theta, T_2(w_t)) \\ \alpha(F_1(v_1, v_{s-1}), F_2(w_1, w_{t-1})) + \alpha(T_1(v_s), T_2(w_t)) \\ c(\lambda, w_t) + \min\limits_{1\leq k\leq s}\{\alpha(F_1(v_1, v_{k-1}), F_2(w_1, w_{t-1})) + \alpha(F_1(v_k, v_s), F_2(w_t))\} \\ c(v_s, \lambda) + \min\limits_{1\leq k\leq t}\{\alpha(F_1(v_1, v_{s-1}), F_2(w_1, w_{k-1})) + \alpha(F_1(v_s), F_2(w_k, w_t))\} \end{cases}$$

Ces deux récursions sont la base d'un algorithme de programmation dynamique.

2.2.1 Une nouvelle récursion pour aligner les arbres et les forêts ordonnés

Nous étudions dans cette section la définition d'une nouvelle récursion qui fusionne les deux récursions précédentes en une seule. L'idée est qu'une forêt est un ensemble ordonné d'arbres et que cet ensemble peut avoir un seul élément. Pour cela, nous proposons cette récursion et nous montrons par la suite son exactitude par rapport à celles de Jiang.

Récursion : Pour $F_1(v_s, v_p)$ et $F_2(w_t, w_q)$ tels que $1 \leq s \leq p \leq i$ et $1 \leq t \leq q \leq j$:

$$\alpha(F_1(v_s, v_p), F_2(w_t, w_q)) =$$
$$min \begin{cases} c(v_p, w_q) + \alpha(F_1(v_p), F_2(w_q)) + \alpha(F_1(v_s, v_{p-1}), F_2(w_t, w_{q-1})) \\ c(\lambda, w_q) + \min\limits_{s\leq k\leq (p+1)}\{\alpha(F_1(v_s, v_{k-1}), F_2(w_t, w_{q-1})) + \alpha(F_1(v_k, v_p), F_2(w_q))\} \\ c(v_p, \lambda) + \min\limits_{t\leq k\leq (q+1)}\{\alpha(F_1(v_s, v_{p-1}), F_2(w_t, w_{k-1})) + \alpha(F_1(v_p), F_2(w_k, w_q))\} \end{cases}$$

Preuve : Considérant un alignement optimal de $F_1(v_s, v_p)$ et $F_2(w_t, w_q)$. Il y a trois cas possibles en raisonnant sur les deux racines *rightmost* des deux forêts F_1 et F_2 :

1. Substitution des deux racines (v_p, w_q). Dans ce cas, le premier cas du min est appliqué.
2. Insertion de la racine *rightmost* de $F_2(w_t, w_q)$, donc nous avons (λ, w_q) et il reste à aligner $F_1(v_s, v_p)$ avec $(F_2(w_t, w_q) - w_q)$, celui là nous le décomposons en deux alignements partiels (1)$F_1(v_s, v_{k-1})$ et $F_2(w_t, w_{q-1})$, (2) $F_1(v_k, v_p)$ et $F_2(w_q)$, tout en cherchant le minimum sur k.
3. Suppression de la racine *rightmost* de $F_1(v_s, v_p)$. Similaire au cas 2.

En appliquant cette récursion aux arbres ($T(v) = F(v, v)$) nous aurons, pour
$T_1(v_p) = F_1(v_p, v_p)$ et $T_2(w_q) = F_2(w_q, w_q)$:

$$\alpha(F_1(v_p, v_p), F_2(w_q, w_q)) =$$

$$\min \begin{cases} c(v_p, w_q) + \alpha(F_1(v_p), F_2(w_q)) + \alpha(F_1(v_p, v_{p-1}), F_2(w_q, w_{q-1})) \\ c(\lambda, w_q) + \min\limits_{p \leq k \leq (p+1)} \{\alpha(F_1(v_p, v_{k-1}), F_2(w_q, w_{q-1})) + \alpha(F_1(v_k, v_p), F_2(w_q))\} \\ c(v_p, \lambda) + \min\limits_{q \leq k \leq (q+1)} \{\alpha(F_1(v_p, v_{p-1}), F_2(w_q, w_{k-1})) + \alpha(F_1(v_p), F_2(w_k, w_q))\} \end{cases}$$

En appliquant $F(v_k, v_{k-1}) = \theta$ et $F(v_k, v_k) = T(v_k)$ tels que $\alpha(\theta, \theta) = 0$, nous aurons :

$$\alpha(T_1(v_p), T_2(w_q)) =$$

$$\min \begin{cases} c(v_p, w_q) + \alpha(F_1(v_p), F_2(w_q)) \\ c(\lambda, w_q) + \alpha(T_1(v_p), F_2(w_q)) \\ c(\lambda, w_q) + \alpha(T_1(v_p), \theta) + \alpha(\theta, F_2(w_q)) \\ c(v_p, \lambda) + \alpha(F_1(v_p), T_2(w_q)) \\ c(v_p, \lambda) + \alpha(\theta, F_2(w_q)) + \alpha(F_1(v_p), \theta) \end{cases}$$

Nous obtenons donc finalement une récursion et un algorithme de programmation dynamique, aussi bien pour l'alignement des forêts que pour celui des arbres.

Cette nouvelle formulation va nous permettre maintenant de nous intéresser aux alignements d'un nombre fini $n > 2$ d'arbres à la fois, puisque que la récursion précédente nous offre la possibilité d'aligner un arbre avec une forêt, ce qui n'était pas possible dans l'algorithme de Jiang.

2.2.2 Alignement multi-arbres

Pour aligner un nombre fini d'arbres, l'idée est de raisonner sur l'étiquette de la racine la plus à droite (*rightmost*) de la forêt résultante, puis de traiter tous les cas que peut avoir cette étiquette (Figure 3). Par exemple, dans l'alignement de deux arbres on a ($2^2 - 1 = 3$) types d'appariement possibles pour la racine, à savoir (v_p, w_q), (λ, w_q) et (v_p, λ).

Considérons maintenant un alignement optimal A^4 de quatre forêts F_1, F_2, F_3 et F_4. Il y a quinze ($2^4 - 1$) étiquettes possibles pour le *rightmost* de A^4 : toutes les combinaisons possibles à partir des *rightmost* des F_i sauf la combinaison $(\lambda, \lambda, \lambda, \lambda)$.

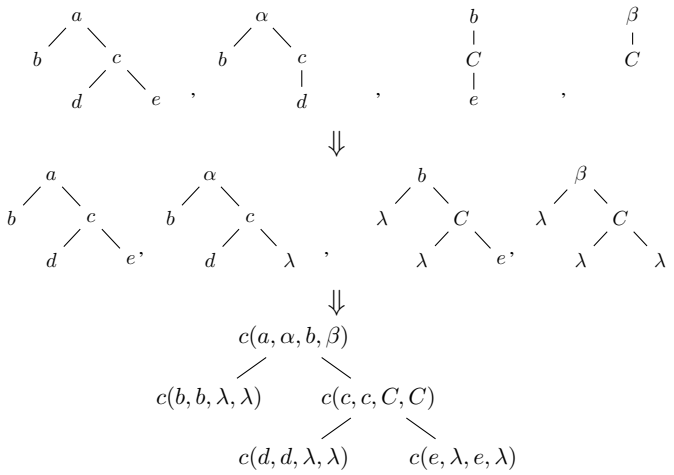


FIG. 3 – Alignement de quatre forêts

L'entrée de notre l'algorithme est alors l'alphabet Σ_λ dans lequel est défini un coût d'appariement sur un quadruplet d'étiquettes. La sortie est le coût d'alignement de quatre arbres.

La récursion suivante forme la base de notre algorithme.

Récursion : Pour tout $\prod$ coût d'alignement et c coût d'appariement d'un quadruplet de nœuds, nous avons :

$$\prod_{F_3[k_r,k_f]\ ,\ F_4[l_p,l_h]}^{F_1[i_s,i_d]\ ,\ F_2[j_t,j_q]} =$$

$$\min \begin{cases} c(\lambda,\lambda,\lambda,l) + \underset{\substack{s\le u\le d+1\\ t\le v\le q+1\\ r\le w\le f+1}}{min} \{\prod_{F_3[k_r,k_{w-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{u-1}]\ ,\ F_2[j_t,j_{v-1}]} + \prod_{F_3[k_w,k_f]\ ,\ F_4[l_h]}^{F_1[i_u,i_d]\ ,\ F_2[j_v,j_q]}\} \\ c(\lambda,\lambda,k,l) + \underset{\substack{s\le u\le d+1\\ t\le v\le q+1}}{min} \{\prod_{F_3[k_r,k_{f-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{u-1}]\ ,\ F_2[j_t,j_{v-1}]} + \prod_{F_3[k_f]\ ,\ F_4[l_h]}^{F_1[i_u,i_d]\ ,\ F_2[j_v,j_q]}\} \\ c(\lambda,j,k,l) + \underset{s\le u\le d+1}{min} \{\prod_{F_3[k_r,k_{f-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{u-1}]\ ,\ F_2[j_t,j_{q-1}]} + \prod_{F_3[k_f]\ ,\ F_4[l_h]}^{F_1[i_u,i_d]\ ,\ F_2[j_q]}\} \\ c(i,j,k,l) + \prod_{F_3[k_r,k_{f-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{d-1}]\ ,\ F_2[j_t,j_{q-1}]} + \prod_{F_3[k_f]\ ,\ F_4[l_h]}^{F_1[i_d]\ ,\ F_2[j_q]} \end{cases}$$

Nous n'avons donné que quatre des quinze quadruplets des étiquettes possibles que peut avoir le *rightmost*. Les autres cas sont similaires. Par exemple, dans le cas où de l'étiquette $(\lambda,\lambda,\lambda,l)$, il est similaire de traiter les cas des étiquettes $(\lambda,\lambda,k,\lambda)$, $(\lambda,j,\lambda,\lambda)$ et $(i,\lambda,\lambda,\lambda)$.

Nous avons montré que la complexité de notre algorithme d'alignement de n arbres est de $O(|T|^n * (\text{ degré } (T))^n)$ en temps et $O(|T|^n * \text{ degré } (T))$ en espace.

La dernière étape consiste alors à déterminer un coût d'appariement sur un quadruplet d'étiquettes dans l'alphabet Σ_λ. La section suivante présente un coût basé sur la notion d'analogie.

3 Analogie et dissimilarité entre arbres

3.1 Apprentissage par proportion analogique

L'analogie, telle que nous l'utiliserons dans ce document, se fonde sur la similitude de termes dans un certain univers pouvant s'énoncer dans une proportion : "A est à B ce que C est à D". Elle peut donc être utilisée, par un mécanisme de mise en correspondance de structures ou de concepts, à transposer des connaissances d'un objet à un autre. Son utilisation en apprentissage se base sur le principe suivant.

Soit un ensemble $\mathcal{S}$ d'objets, définis par un ensemble de valeurs des attributs c_1, c_2,...,c_n, prenant leurs valeurs dans des domaines éventuellement différents. Soit x un nouvel objet dont seule une partie des attributs $c_1, ..., c_n$ est connue. Une inférence analogique détermine les caractéristiques inconnues de x en suivant un schéma en deux étapes :

1. toutes les proportions analogiques mettant en jeu les attributs connus de x et les attributs correspondants de trois objets de $\mathcal{S}$ sont identifiées ;
2. pour chacun des triplets d'objets de $\mathcal{S}$ identifiés, le transfert analogique est calculé pour obtenir les attributs inconnus de x. Si le transfert réussit, son résultat est ajouté aux solutions possibles.

En particulier, si le seul attribut inconnu de x est sa classe, on se retrouve dans un problème d'apprentissage de règle de classification, avec une méthode « paresseuse ». Les références [15, 12] montrent comment rendre opérationnel ce schéma, en évitant en particulier d'examiner tous les triplets de $\mathcal{S}$ et/ou en utilisant la notion de dissimilarité analogique.

3.2 Analogie et proportion analogique

D'une façon formelle, [10] introduit la notion de proportion analogique entre quatre objets comme :

Définition 3.1 *Une proportion analogique sur un ensemble X est une relation sur X^4 qui respecte trois axiomes. Nous notons quatre éléments $(A,B,C,D) \in X$ en proportion analogique, par : $A : B :: C : D$, ce qui se lit "A est à B ce que C est à D". Les axiomes sont les suivants :*

– *Symétrie de la relation "'est à"' : $A : B :: C : D \Leftrightarrow C : D :: A : B$.*

– *Échange des moyens : $A : B :: C : D \Leftrightarrow A : C :: B : D$.*

– *Déterminisme :* $A : B :: A : x \Rightarrow x = B$ *et* $A : A :: B : x \Rightarrow x = B$.

Dans [15], les auteurs ont proposé une définition générale de la proportion analogique, qui s'appuie sur un cadre algébrique générique et se décline pour un grand nombre de structures telles que les séquences et les arbres étiquetés. L'idée est de décomposer les quatre objets en question sous forme d'une factorisation.

Nous, en s'inspirant des travaux faits sur les séquences dans [12], nous aboutissons à la nouvelle définition suivante tout en gardant la cohérence des axiomes définies précédemment :

Définition 3.2 *Soit x, y, z et t quatre arbres d'étiquettes dans Σ. On suppose qu'une analogie dans Σ_λ est définie. Nous disons que ces arbres sont en proportion analogique s'il existe un alignement des quatre arbres x', y', z' et t' d'étiquettes $\in \Sigma_\lambda$, vérifiant :*

– $\forall\ i$ *une position d'un nœud, la proportion analogique sur les nœuds* $x'_i : y'_i :: z'_i : t'_i$ *est vraie.*

Les propriétés posées par Zhang dans l'alignement de deux arbres portent sur les appariements possibles des paires de nœuds. Nous conservons ces conditions et nous l'étendons à des quadruplets des nœuds qui forment les proportions analogiques construites par alignement.

Propriété 3.1 *Pour tous les proportions analogiques* $(x_{i'_1} : y_{i'_1} :: z_{i'_1} : t_{i'_1})$ *et* $(x_{i'_2} : y_{i'_2} :: z_{i'_2} : t_{i'_2})$ *dont les constituants* $\in \Sigma$,

1. $x_{i'_1} = x_{i'_2}$ *dans* $x \Longleftrightarrow \begin{cases} y_{i'_1} = y_{i'_2} \text{ dans } y, \\ z_{i'_1} = z_{i'_2} \text{ dans } z, \\ t_{i'_1} = t_{i'_2} \text{ dans } t. \end{cases}$

2. $x_{i'_1}$ *est à gauche de* $x_{i'_2}$ *dans* $x \Longleftrightarrow \begin{cases} y_{i'_1} \text{ est à gauche de } y_{i'_2} \text{ dans } y, \\ z_{i'_1} \text{ est à gauche de } z_{i'_2} \text{ dans } z, \\ t_{i'_1} \text{ est à gauche de } t_{i'_2} \text{ dans } t. \end{cases}$

3. $x_{i'_1}$ *est un ancêtre de* $x_{i'_2}$ *dans* $x \Longleftrightarrow \begin{cases} y_{i'_1} \text{ est un ancêtre de } y_{i'_2} \text{ dans } y, \\ z_{i'_1} \text{ est un ancêtre de } z_{i'_2} \text{ dans } z, \\ t_{i'_1} \text{ est un ancêtre de } t_{i'_2} \text{ dans } t. \end{cases}$

Ces conditions nécessaires non suffisantes[4] permettent la conservation de structures des arbres lors de l'alignement. En effet, le premier point indique qu'un nœud n'est concerné que par une et une seule proportion. Le point 2 détermine la conservation de l'ordre gauche-droite des nœuds lors de l'alignement des quatre arbres. Le point 3 traduit le maintien de l'ordre entre ancêtre-descendant.

3.3 Calcul et validation de la proportion analogique

Pour mettre en œuvre notre définition nous proposons un algorithme polynomial, tout en introduisant une nouvelle notion : la *dissemblance ou dissimilarité analogique* entre arbres.

3.3.1 Dissimilarité analogique entre arbres

Dans cette section, nous définissons une proportion analogique approximative, que pourrait traduire l'expression linguistique *« a est à b à peu prés ce que c est à d »*. Nous mesurons le terme « à peu près » par une certaine valeur réelle positive, égale à 0 quand l'analogie est exacte, et croissante au fur et à mesure que les quatre objets sont de moins en moins en proportion. nous appelons cette valeur *dissemblance analogique (DA)*.

Cette mesure a été introduit aux séquences par [12], nous l'étendons ici aux arbres.

[4]Une quatrième condition est en cours de vérification pour caractériser l'alignement comme distance d'édition entre arbres sous contraintes.

Dissemblance analogique entre arbres Nous présentons dans ce qui suit la mise en œuvre de notre propre définition de proportion analogique entre arbres et définissons la notion de *dissemblance analogique DA* ainsi que ses propriétés par rapport à l'analogie.

Nous supposons qu'il existe dans l'alphabet Σ_λ une dissemblance analogique DA, c'est à dire une relation telle que pour tout $a \in \Sigma$ et pour tous $b, c, d \in \Sigma_\lambda$, on ait : $DA(a, a, b, b) = 0$, $DA(a, a, a, a) = 0$, et $DA(a, b, c, d) > 0$.

La dissemblance analogique entre arbres ordonnés et étiquetés est alors définie par :

Définition 3.3 *Soit x, y, z et t quatre arbres d'étiquettes dans Σ_λ. La dissemblance analogique $DA(x, y, z, t)$ est le coût minimal d'un alignement des quatre arbres.*

Propriété 3.2 *Cette définition assure que les propriétés suivantes restent vraies pour x, y, z et t quatre arbres.*

Cohérence avec l'analogie .
$DA(x, y, z, t) = 0 \Leftrightarrow x : y :: z : t.$

Symétrie de la relation *"à peu prés ce que"* et échange de moyens .
$DA(x, y, z, t) = DA(z, t, x, y) = DA(x, z, y, t).$

Non Symétrie de la relation *"est à"* .
$DA(x, y, z, t) = DA(y, x, z, t)$

Le calcul de la dissimilarité analogique entre arbres est ainsi ramené à l'utilisation de la technique d'alignement multi-arbre déjà défini.

Calcul de la *DA* entre arbres : algorithme *AnaTree* En s'inspirant de l'idée derrière l'alignement de deux arbres, nous proposons dans ce qui suit un algorithme de programmation dynamique appelé *AnaTree* pour construire un alignement analogique optimal entre quatre arbres et par la suite calculer la *DA* entre ces arbres comme le coût minimum de cet alignement.

L'entrée de l'algorithme est l'alphabet Σ_λ dans lequel est définie une dissemblance analogique sur les étiquettes. La sortie de cet algorithme est la DA entre quatre arbres.

La récursion suivante forme la base de notre algorithme.

Récursion :

Pour tout $\prod$ coût de DA entre arbre et DA coût de dissemblance analogique sur les étiquettes nous avons :

$$\prod_{F_3[k_r,k_f]\ ,\ F_4[l_p,l_h]}^{F_1[i_s,i_d]\ ,\ F_2[j_t,j_q]} =$$

$$\min \begin{cases} DA(\lambda,\lambda,\lambda,l) + \min\limits_{\substack{s\le u\le d+1\\ t\le v\le q+1\\ r\le w\le f+1}} \{\prod_{F_3[k_r,k_{w-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{u-1}]\ ,\ F_2[j_t,j_{v-1}]} + \prod_{F_3[k_w,k_f]\ ,\ F_4[l_h]}^{F_1[i_u,i_d]\ ,\ F_2[j_v,j_q]}\} \\ DA(\lambda,\lambda,k,l) + \min\limits_{\substack{s\le u\le d+1\\ t\le v\le q+1}} \{\prod_{F_3[k_r,k_{f-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{u-1}]\ ,\ F_2[j_t,j_{v-1}]} + \prod_{F_3[k_f]\ ,\ F_4[l_h]}^{F_1[i_u,i_d]\ ,\ F_2[j_v,j_q]}\} \\ DA(\lambda,j,k,l) + \min\limits_{s\le u\le d+1} \{\prod_{F_3[k_r,k_{f-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{u-1}]\ ,\ F_2[j_t,j_{q-1}]} + \prod_{F_3[k_f]\ ,\ F_4[l_h]}^{F_1[i_u,i_d]\ ,\ F_2[j_q]}\} \\ DA(i,j,k,l) + \prod_{F_3[k_r,k_{f-1}]\ ,\ F_4[l_p,l_{h-1}]}^{F_1[i_s,i_{d-1}]\ ,\ F_2[j_t,j_{q-1}]} + \prod_{F_3[k_f]\ ,\ F_4[l_h]}^{F_1[i_d]\ ,\ F_2[j_q]} \end{cases}$$

4 Conclusion

Dans cet article, nous nous sommes intéressés à la définition d'une proportion analogique entre arbres étiquetés et ordonnés, nous avons défini cette notion conformément aux axiomes classiques puis nous l'avons étendue au concept de dissimilarité analogique, qui a déjà été utilisé en apprentissage pour les objets structurés en séquences. Notre but est d'appliquer les méthodes d'apprentissage par analogie à de tels arbres, qui codent en particulier les structures prosodiques des phrases, pour la synthèse de la parole. D'autres applications sont envisageables dans les documents structurés (XML), la bio-informatique (les structures secondaires d'ARN), etc. L'étape suivante de ces travaux sera la mise en œuvre d'algorithmes analogiques rapides pour l'apprentissage et la génération d'arbres.

Références

[1] A. Aamodt et E. Plaza. Case-based reasoning : Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1) :39–59, 1994.

[2] S. Bayoudh, H. Mouchère, L. Miclet, et E. Anquetil. Learning a classifier with very few examples : analogy based and knowledge based generation of new examples for character recognition. In *Proc. of ECML*, volume 18, pages 527–534. Springer Verlag LNAI 4701, 2007.

[3] Laurent Blin. Prosody prediction from tree-like structure similarities. In *TSD*, pages 369–374, 2000.

[4] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, et M. Tommasi. Tree automata techniques and applications. Available on : http ://www.grappa.univ-lille3.fr/tata, 2007. release October, 12th 2007.

[5] Erik D. Demaine, Shay Mozes, Benjamin Rossman, et Oren Weimann. An optimal decomposition algorithm for tree edit distance. In *ICALP*, pages 146–157, 2007.

[6] Serge Dulucq et Hélène Touzet. Analysis of tree edit distance algorithms. In *In Proceedings of the 14th annual symposium on Combinatorial Pattern Matching (CPM*, pages 83–95. Springer-Verlag, 2003.

[7] T. Evans. A heuristic program to solve geometry analogy problems. In *Semantic Information Processing*. MIT Press, Cambridge, 1968.

[8] K. Holyoak. Analogy. In *The Cambridge Handbook of Thinking and Reasoning*, chapter 6. Cambridge University Press, 2005.

[9] Philip N. Klein. Computing the edit-distance between unrooted ordered trees. In *ESA*, pages 91–102, 1998.

[10] Y. Lepage. *De l'analogie rendant compte de la commutation en linguistique*. Grenoble, 2003. Habilitation à diriger les recherches.

[11] Chin Lung Lu, Zheng-Yao Su, et Chuan Yi Tang. A new measure of edit distance between labeled trees. In *COCOON*, pages 338–348, 2001.

[12] L. Miclet, S. Bayoudh, et A. Delhay. Analogical dissimilarity : Definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research*, 32 :793–824, 2008.

[13] Thorsten Richter. A new measure of the distance between ordered trees and its applications, technical report 85166-cs. *Technical report, Department of Computer Science, University of Bonn*, 1997.

[14] N. Stroppa et F. Yvon. Analogical learning and formal proportions : Definitions and methodological issues. Technical Report ENST-2005-D004, École Nationale Supérieure des Télécommunications, June 2005.

[15] N. Stroppa et F. Yvon. Apprentissage par analogie et proportions formelles : contributions méthodologiques et expérimentales. In F. Denis, editor, *CAp 2005*. PUG, 2005.

[16] Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3) :422–433, 1979.

[17] Lusheng Wang Tao Jiang et Kaizhong Zhang. Alignment of trees - an alternative to tree edit. In *CPM '94 : Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, pages 75–86, London, UK, 1994. Springer-Verlag.

[18] Robert A. Wagner et Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1) :168–173, 1974.

[19] Kaizhong Zhang. Algorithms for the constrained editing distance between ordered labeled trees and related problems. *Pattern Recognition*, 28(3) :463–474, 1995.

[20] Kaizhong Zhang et Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6) :1245–1262, 1989.

Étendre les possibilités du raisonnement à partir de cas grâce aux traces

Amélie Cordier, Bruno Mascret, Alain Mille
Université de Lyon,
CNRS, Université Lyon 1,
LIRIS, UMR5205, F-69622, France
{firstname.lastname}@liris.cnrs.fr

Résumé

Dans le domaine de l'acquisition des connaissances, l'un des objectifs en vogue est de construire des systèmes informatiques permettant le partage et la réutilisation d'expériences au sein de vastes communautés d'utilisateurs. Afin d'être utilisables, de tels systèmes doivent faire preuve d'une certaine plasticité et proposer des interfaces de consultation agréables et intuitives. Les outils de raisonnement à partir de cas, de par leurs propriétés intrinsèques, semblent être de bons candidats pour l'assistance au partage et à la réutilisation de l'expérience et pourtant, une de leurs limites actuelles est justement leur manque de flexibilité. En effet, ces outils et ces systèmes sont souvent développés afin de résoudre un problème particulier et ne sont donc pas conçus pour être modifiés par les utilisateurs en fonction de l'usage qu'ils souhaitent en faire. Dans cet article, nous proposons d'utiliser des objets appelés "traces d'interactions" comme des sources de connaissances pour le RÀPC. Ces traces d'interactions témoignent de l'utilisation que les utilisateurs font des systèmes et enregistrent, dans un contexte riche, chaque expérience de résolution de problème. Nous pensons que l'usage des traces comme support du raisonnement va permettre de repousser les limites actuelles du raisonnement à partir de cas et nous illustrons cette hypothèse sur un bref exemple.

Mots clés : Raisonnement à partir de cas (RÀPC), traces, interactions homme-machine, contexte, aides techniques pour le handicap,.

1 Introduction

Pour résoudre un nouveau problème, le raisonnement à partir de cas s'appuie sur l'exploitation d'une mémoire d'expériences passées (des épisodes de résolution de problèmes) plutôt que sur l'utilisation d'un ensemble de règles. Cette approche a des avantages majeurs bien connus : les systèmes de RÀPC ne construisent pas des solutions à partir de rien puisqu'ils peuvent s'appuyer sur des cas passés. Par conséquent, ils n'ont pas besoin de disposer de la formalisation complète de la théorie d'un domaine pour commencer à raisonner : ils démarrent avec très peu de connaissances et sont capables d'apprendre et de s'améliorer avec chaque épisode de résolution de problème. Pour toutes ces raisons, le RÀPC est considéré comme un puissant paradigme de raisonnement facile à mettre en œuvre.

Cependant, malgré ces avantages, le RÀPC souffre de problèmes de ré-ingénierie : faire en sorte qu'un système de RÀPC évolue est toujours très difficile. Par exemple, les cas sont des vues instantanées qui capturent des expériences passées dans une structure figée qui n'est en général pas conçue pour évoluer. Si l'on souhaite faire évoluer la représentation d'une expérience, par exemple pour mieux prendre en compte son contexte, les contraintes de la structure du cas nous limitent très souvent. Ce manque de flexibilité de la représentation des connaissances est sans aucun doute une véritable limite du RÀPC.

Nous pensons qu'il est possible d'accroître considérablement les possibilités du RÀPC en développant des systèmes plus malléables et nous sommes convaincus qu'une approche s'appuyant sur les

traces constitue un début de réponse possible. En effet, les traces d'interactions, utilisées comme des enregistrements continus d'une activité, permettent de conserver naturellement et en permanence des expériences de résolution de problèmes "contextualisées".

Dans cet article, nous montrons comment une approche à base de traces (que nous introduisons dans la section 3) peut considérablement améliorer les possibilités du RÀPC. La section 4 montre comment les traces d'interactions et le paradigme du RÀPC peuvent fusionner en une architecture générale de RÀPC basée sur les traces. Nous illustrons les bénéfices de cette approche par un exemple théorique dans lequel les traces d'interaction permettent de dépasser une des limites du RÀPC classique. La section 5 donne un aperçu de certains défis ouverts par la proposition de notre approche et discute des perspectives de recherche envisageables. La section 6 étudie les applications possibles de ce travail dans plusieurs domaines couvrant une large gamme de problèmes aux objectifs différents. Enfin, nous concluons par une discussion sur cette contribution dans la section 7.

2 Raisonnement à partir de cas

Le succès du RÀPC s'explique en partie par le fait qu'il constitue une solution, au moins partielle, au problème de l'acquisition des connaissances dans les systèmes à base de connaissances. En effet, les systèmes de RÀPC acquièrent de nouvelles connaissances en accumulant des cas représentant des expériences de résolution de problèmes et les mémorisent d'une façon prédéfinie de sorte qu'ils puissent être réutilisés pour résoudre des problèmes futurs. Les cas constituent donc le principal conteneur de connaissances du RÀPC.

Un cas est habituellement formé d'une partie problème et d'une partie solution, chaque partie étant composée d'un ensemble de descripteurs, la plupart du temps définis dans une ontologie dédiée. Un nouveau problème est appelé un *cas cible*. Les cas sont comparés sur la base de leurs parties problèmes : la description de la partie problème d'un cas cible est comparée à celle de la partie problème de chacun des *cas sources* disponibles dans la base de cas. Les différences observées entre les descripteurs de problèmes sont alors exploitées par des opérateurs d'adaptation chargés de l'adaptation des descripteurs de solutions. L'adaptation de la solution réalisée pendant cette phase produit une solution candidate pour le cas cible.

Les connaissances de similarité, utilisées pour comparer les parties problèmes, sont également utilisées comme connaissances d'adaptation [6]. Lorsqu'une solution candidate ne fonctionne pas dans le contexte du cas cible, il est possible de réviser le cas (en s'appuyant sur des connaissances externes) et, en conséquence, de réviser les connaissances du système. Les cas, révisés ou non, sont toujours stockés dans la base de cas une fois qu'ils ont été résolus, en vue d'une remobilisation future.

Le RÀPC est un bon système à base de connaissances, qui fonctionne plutôt bien et qui remporte un important succès industriel [12], mais son ingénierie reste un problème difficile. Par exemple, les opérations de maintenance sont fastidieuses et requièrent d'importants efforts de ré-ingénierie des connaissances. Les réponses classiques pour améliorer l'ingénierie du RÀPC sont en fait les mêmes que celles utilisées dans le génie logiciel traditionnel [4]. Ainsi, en dépit de ses nombreuses qualités, le RÀPC n'est pas vraiment un système à base de connaissances qui "apprend à résoudre des problèmes en résolvant des problèmes" puisqu'il n'est pas capable de faire évoluer dynamiquement sa propre représentation des expériences. Nous pouvons identifier trois causes principales à cette limitation :

- le modèle de cas **"trop bien structuré et donc trop contraint"** : un cas doit être complètement décrit, le plus souvent selon une structure statique et rigide, ce qui est un frein à toute évolutivité. Cette limitation est une variante du traditionnel "frame problem". Elle réduit de manière drastique le domaine de compétence du RÀPC en empêchant toute résolution de problème "non prévu" ;
- le paradoxe des **"connaissances figées qui évoluent"** : les connaissances d'un système de RÀPC sont supposées cohérentes ou invariantes à chaque instant (cas résolus, connaissances du domaine, similarité et connaissances d'adaptation). Dans une telle situation, comment permettre l'évolution des connaissances du système tout en maintenant leur cohérence, et sans trop solliciter l'ingénieur de la connaissance ?

– l'hypothèse réductrice du **"c'est bon pour moi maintenant donc ce sera toujours bon pour tout le monde"** : dans les applications non triviales, il est fréquent que la question que l'on souhaite poser change radicalement au fur et à mesure des utilisations. Toutefois, les systèmes de RÀPC sont habituellement conçus pour résoudre un problème bien particulier. Comment est-il alors possible de créer un système capable de résoudre plusieurs problèmes, même lorsque ceux-ci n'ont pas été anticipés par le concepteur ?

Le débat est donc ouvert : comment concevoir un système de RÀPC capable de prendre en compte un contexte non connu, en évolution permanente et qui ne nécessite pas une ré-ingénierie coûteuse ? Comme il est illusoire de chercher à anticiper tous les usages possibles d'un système, comment concevoir des systèmes capables de s'adapter eux-mêmes à leurs utilisateurs ? Afin de dépasser ces limites, est-il possible d'associer apprentissage et résolution de problème dans un cycle de raisonnement revisité ?

Notre proposition est d'aller plus loin que la simple collecte d'épisodes de résolution de problèmes et de considérer les traces d'interactions comme un conteneur de connaissances dans lequel plusieurs situations de résolution de problèmes diversifiées peuvent être retrouvées. Nous proposons une variante étendue, en terme de représentation des connaissances, du RÀPC que nous appelons raisonnement à partir de l'expérience tracée (RÀPET). Dans le RÀPET, le raisonnement est toujours basé sur les cas, mais ceux-ci sont "construits" dynamiquement à partir de la trace, dans la perspective précise du raisonnement. Il ne sont plus stockés dans la base de cas mais sont présents dans le contexte de la trace d'activité. Le cycle du RÀPET s'appuie sur la notion de trace d'interactions, en tant qu'enregistrement d'une activité représentée à un niveau d'abstraction approprié. Les traces sont elles-mêmes gérées par un système de gestion des traces (voir la section 3 pour ceux qui n'auraient jamais entendu parler du TBMS).

Pendant la phase d'**élaboration**, une requête est construite. Cette requête va impliquer une étape de réutilisation d'expérience. Elle correspond à une *signature expliquée de cas*, c'est-à-dire à un motif caractérisant le problème qui doit être résolu. Ce motif est similaire à la définition de cas cible en RÀPC. L'élaboration est guidée par les connaissances du système. Il est parfois nécessaire de transformer la trace de sorte qu'elle soit décrite avec le vocabulaire disponible dans l'ontologie et utilisée pour exprimer classiquement la requête. La notion de transformation de trace est essentielle (voir section 3). La requête décrit la partie problème et la signature expliquée de cas apporte des précisions sur les descripteurs de solution attendus. Par conséquent, elle donne également les contraintes requises pour permettre la fouille et la découverte de motifs similaires. En RÀPC, l'étape d'élaboration est souvent limitée à un remplissage de formulaire ou à un processus similaire. En RÀPET, cette étape joue un rôle fondamental et fournit les connaissances qui permettront de guider efficacement le raisonnement. Des connaissances supplémentaires peuvent être ajoutées par l'utilisateur en fonction du contexte précis de sa requête (signatures de cas spécifiques ou mesures de similarité spécifiques par exemple).

D'après la requête posée (c'est-à-dire le cas cible), plusieurs épisodes retrouvés dans la trace sont considérés comme similaires à la signature élaborée dans la requête. La similarité est calculée en utilisant une mesure qui prend en compte les connaissances d'adaptation disponibles. À cette mesure est associée la signature du cas qui a éventuellement été spécialisée durant la phase d'élaboration. Le processus de **remémoration** peut être interrompu à tout moment afin d'améliorer la définition de la requête et/ou de raffiner la mesure de similarité. Ceci revient à faire une nouvelle élaboration et donc à apprendre d'après les échecs du processus de remémoration.

Lorsque l'utilisateur (ou le système) décide d'utiliser un épisode précédent, cet épisode est adapté exactement comme un cas serait adapté en RÀPC. Les règles d'**adaptation** sont associées à chaque signature expliquée de cas. Si l'adaptation échoue, les connaissances d'adaptation peuvent être corrigées durant la phase de **révision**. Il est souvent nécessaire de mettre en place une boucle sur l'étape d'élaboration afin d'enrichir le contexte de la requête. Le cas révisé est ensuite proposé à l'utilisateur, et bien entendu enregistré dans la trace courante.

En RÀPET, il n'y a pas d'étape spécifique dédiée à l'**apprentissage** puisque celui-ci est au cœur du cycle. Cependant, l'apprentissage au sens de la mémorisation des nouvelles connaissances construites

durant l'épisode de résolution de problème) se concrétise plus particulièrement lors des étapes d'élaboration et de révision.

3 Traces

Une "trace" peut être définie comme un enregistrement de "quelque chose" qui s'est déroulé dans le passé. Une trace est une empreinte, c'est-à-dire ce qu'il reste d'un phénomène après son déroulement. En informatique, les traces sont partout : fichiers de log, historiques de navigation, gestion de versions, etc. Elles ont également été étudiées pour différentes applications [9]. Récemment, plusieurs recherches ont contribué à l'élaboration d'une *théorie de la trace* [17]. D'après cette théorie, une trace est un ensemble d'éléments situés à la fois temporellement et spatialement et qui sont inscrits dans l'environnement pendant l'activité. Les traces peuvent résulter d'inscriptions intentionnelles ou non.

La théorie de la trace, ainsi que les travaux qui y sont associés, fournissent tous les éléments nécessaires à l'exploitation des traces numériques : vocabulaire, méthodes et outils pour les manipuler. Selon cette approche, les traces sont gérées au sein d'un système à base de traces (TBMS) qui fournit les méthodes pour collecter, transformer, stocker, manipuler et visualiser les traces. Ces méthodes permettent de s'attaquer à un problème majeur : comment construire des traces (en collectant un ensemble d'objets pertinents) qui seront réutilisables dans un but particulier ? Cela dit, seule la moitié du chemin est faite puisque les-dites méthodes doivent tout de même s'appuyer sur des modèles qui doivent être définis en fonction de l'application visée. Afin de s'affranchir de ce problème, la théorie de la trace définit donc le concept de `M-trace` (plus connue sous le nom de trace modélisée), qui permet de considérer les traces à un certain niveau d'abstraction et en prenant en compte l'usage que l'on souhaite en faire. Le concept de `M-trace` témoigne de l'indivisibilité de la trace et de son modèle. Les `M-traces` peuvent être utilisées et traitées grâce aux outils de transformation qui permettent de filtrer, fusionner et même reformuler des traces.

Les traces sont des ensembles d'objets collectés à partir de ce que l'on appelle une trace primaire, c'est-à-dire une source de données brute. Il est alors possible de retrouver dans ces traces des "épisodes" (qui sont des enregistrements de situations précédentes) en s'appuyant sur des méthodes de manipulation et de transformation des traces. Sans ces transformations et une reformulation en `M-trace`, une trace première ne peut servir efficacement de support pour le raisonnement (figure 1). Ces transformations utilisent au départ des opérateurs standards (suppression, fusion, insertion, etc.) et s'appuient sur :

- les connaissances du domaine ;
- les connaissances sur la manière dont un utilisateur interagit avec ce domaine ;
- les transformations déjà appliquées dans un contexte similaire ;
- l'objectif actuel (l'intention) du processus de RÀPET.

Le TBMS est un outil dédié à la gestion des traces et il peut être utilisé dans différents buts. Dans la suite, nous cherchons à utiliser les fonctionnalités du TBMS afin d'accroître les possibilités du RÀPC. Nous avons la conviction qu'introduire le concept de traces d'interactions dans le RÀPC est une solution "pas si compliquée" pour mieux prendre en compte le contexte lors de la résolution de problème et donc pour faciliter la réutilisation de l'expérience.

4 Étendre le RÀPC avec les traces : un exemple tiré du chapeau

Cette section décrit les avantages de l'approche que nous proposons en s'appuyant sur un exemple théorique issu d'un domaine d'application fictif, mais néanmoins connus des spécialistes : *FLATLAND*. Cet exemple s'inspire d'une nouvelle d'Edwin A. Abbott [1], un des pionnier de la science fiction. Une version élaborée du monde de *FLATLAND* transposé dans les dimensions du CBR a été développée dans [7] afin d'illustrer l'approche IAKA[1] puis reprise et enrichie dans [14].

[1] InterActive Knowledge Acquisition

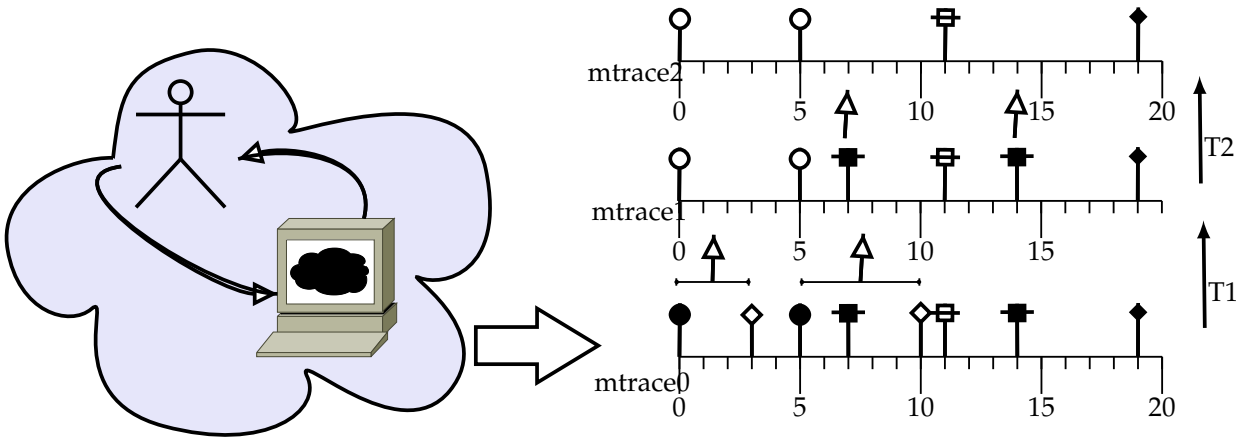


FIG. 1 – De la trace primaire à la M-trace : cette figure décrit la construction d'une $\texttt{M-trace}_2$. Les interactions de l'utilisateur et les objets du domaine sont d'abord collectés, afin d'élaborer une trace première ($\texttt{M-trace}_0$). Cette trace première est alors composée d'observés temporellement situés. La trace première est ensuite reformulée par les transformations T_1 (règle de fusion produisant $\texttt{M-trace}_1$), puis T_2 (règles de suppression conduisant à $\texttt{M-trace}_2$). Ces opérations produisent au final une trace suffisament abstraite (ou de niveau élevé) pour permettre aux processus de raisonnement de s'appliquer plus facilement. Remarquons que ces transformations doivent être réversibles et qu'une M-trace a donc besoin de savoir comment elle a été produite (c'est-à-dire connaître sa provenance).

Dans ce domaine, les problèmes sont des paires ordonnées de formes géométriques, et les solutions sont également des formes (en général, une seule). Pour simplifier l'exemple, nous considérons que les formes n'ont que deux propriétés : leur nombre de côtés et leur couleur. Contrairement aux apparences, il n'existe aucune règle permettant à un quelconque système d'inférer simplement la solution d'un problème étant donnée sa description.

Dans l'approche IAKA, l'adaptation repose sur un processus différentiel décomposable. Ce processus est illustré sur la figure 2 : chaque différence (ou similarité ?) $\texttt{r}_i$ entre un problème source src et un problème cible cible est interprétée et associée à un opérateur d'adaptation spécifique $(\texttt{r}_i, \mathcal{A}_{\texttt{r}_i})$ afin de produire, pas à pas, une solution candidate ($\widetilde{\texttt{Sol}}(\texttt{pb}_1)$ ou $\widetilde{\texttt{Sol}}(\texttt{cible})$) en s'appuyant sur la solution intermédiaire précédente (resp. $\texttt{Sol}(\texttt{srce})$ ou $\widetilde{\texttt{Sol}}(\texttt{pb}_1)$). En d'autres termes, l'approche IAKA lie directement et explicitement les connaissances de similarité aux connaissances d'adaptation au sein même du processus de RÀPC.

Dans le domaine de *FLAT LAND*, plusieurs comportements peuvent être représentés. Le comportement illustré sur la figure 2 est celui que nous appelons "avoir un enfant". Dans ce domaine, la solution (ici, l'enfant) se calcule en suivant deux règles simples : la solution a un côté de plus que la première forme parente et la couleur de la seconde. Ces règles sont données uniquement à titre indicatif et pour une meilleure compréhension, mais le système de RÀPC n'a ni la connaissance des règles ni celle de l'existence de différents comportements possibles. En revanche, le système de RÀPC que nous utilisons ici est capable de représenter les connaissances relatives au comportement "avoir un enfant" en s'appuyant sur des connaissances de similarité et des connaissances d'adaptation.

Pour compliquer les choses, supposons maintenant que d'autres comportements puissent être modélisés dans *FLAT LAND* mais qu'ils n'ont pas encore été implémentés, ni même imaginés, par le concepteur du système. Par exemple, considérons l'exemple du "combat". Les règles associées à un comportement grégaire diffèrent de celles associées au comportement "avoir un enfant" : dans un combat, la première forme gagne le combat mais perd un côté. La figure 3 montre le chemin d'adaptation qui doit être suivi afin de produire une solution candidate pertinente. Le problème qui se pose est lié au fait que le comportement étudié n'a pas été anticipé et que, par conséquent, les connaissances nécessaires au raisonnement ne sont pas forcément disponibles. La réponse classique à

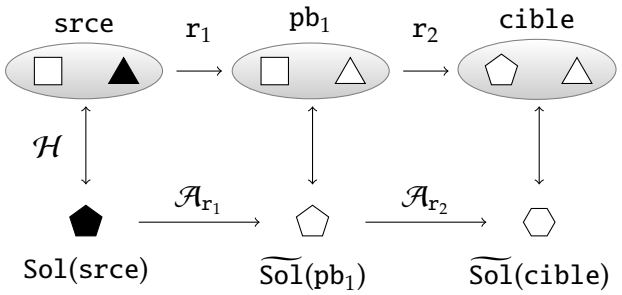


FIG. 2 – Un chemin d'adaptation dans IAKA. La relation $\mathcal{H}$ (verticale) représente le lien entre la partie problème (en haut) et la partie solution (en bas). La première étape prend en compte la différence $\mathtt{r}_1$ (la seconde forme a une couleur différente) : elle propose une solution candidate intermédiaire $\widetilde{\mathtt{Sol}}(\mathtt{pb}_1)$ en adaptant la solution du cas source $\mathtt{Sol}(\mathtt{srce})$ grâce à la fonction d'adaptation $\mathcal{A}_{\mathtt{r}_1}$ (changer la couleur de la solution). La paire $(\mathtt{r}_1, \mathcal{A}_{\mathtt{r}_1})$ est appelée un opérateur d'adaptation. L'étape suivante repose sur le même principe : $\mathtt{r}_2$ (un côté de plus sur la première forme) implique $\mathcal{A}_{\mathtt{r}_2}$ (un côté de plus sur la solution).

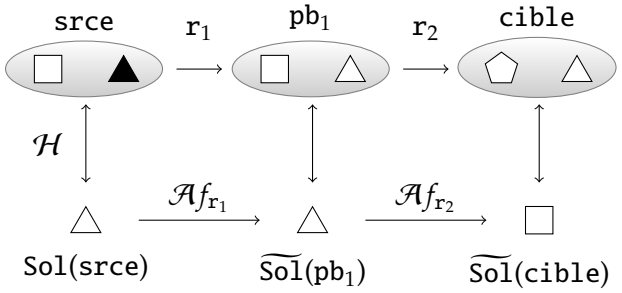


FIG. 3 – Un chemin d'adaptation pour le comportement "combat". Les problèmes sources et cible sont les mêmes que dans le cas précédent (figure 2). Cependant, les solutions diffèrent du comportement "avoir un enfant". L'opérateur d'adaptation $(\mathtt{r}_1, \mathcal{A}f_{\mathtt{r}_1})$ implique qu'une différence de couleur entre les deuxièmes formes ne change pas la couleur de la solution. Les opérateurs $(\mathtt{r}_2, \mathcal{A}f_{\mathtt{r}_2})$ et $(\mathtt{r}_2, \mathcal{A}_{\mathtt{r}_2})$ sont eux identiques.

ce genre de problème, en RÀPC, est de refaire un cycle complet d'ingénierie de la connaissance afin d'acquérir de nouvelles connaissances du domaine. En fonction des modèles et des méthodes utilisés, plusieurs types non exclusifs de connaissances peuvent être acquis : nouvelles définitions de problèmes, nouvelles connaissances de similarité, nouvelles mesures de similarités, nouvelles connaissances d'adaptation. Ces opérations sont coûteuses, souvent très longues et requièrent l'implication forte des concepteurs de systèmes et des experts du domaine. L'utilisateur final doit quant à lui attendre que les améliorations apportées aux connaissances soient propagées jusqu'à lui, et espérer ne pas rencontrer à nouveau de comportements inattendus.

Cependant, si l'on analyse la situation à un instant donné, les connaissances disponibles ne sont pas si mauvaises : elles sont juste partielles et manquent de contexte. Du point de vue de l'utilisateur final, qui n'a d'autre choix que d'attendre qu'on lui apporte la solution, la situation est frustrante. En effet, l'utilisateur sait peut-être *pourquoi* le processus de RÀPC a échoué. Dans ce cas, pourquoi ne pas lui offrir la possibilité d'expliquer au système la cause de l'échec ? Nous allons montrer que cette explication peut se faire simplement en relâchant certaines contraintes d'analyse et en permettant une interprétation plus large de la situation.

L'hypothèse fondamentale du RÀPC est que "des problèmes similaires ont des solutions similaires". Les problèmes du type "avoir un enfant" sont différents des problèmes de "combat" mais, pour l'instant,

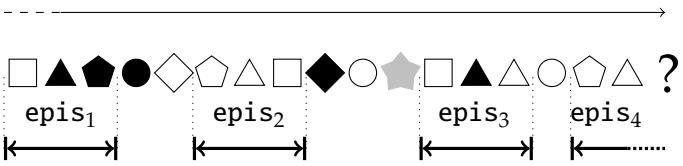


FIG. 4 – Une trace avec un changement de contexte : l'étoile grise indique une marque de contexte qui peut être utilisée pour résoudre une ambiguïté dans le processus d'adaptation pour calculer la solution de l'épisode 4.

leurs descriptions sont identiques. Que se passerait-il si l'on admettait que des problèmes peuvent avoir des solutions différentes en fonction du contexte dans lequel on les considère ?

Dans ce qui suit, nous avons permis à notre système de RÀPC fictif de proposer plusieurs solutions pour un même problème en fonction des contextes correspondants. Le système doit juste pouvoir demander à l'utilisateur dans quel contexte il doit se placer pour raisonner. Grâce à cette information, une solution candidate fiable peut être élaborée. Dans nos deux exemples précédents, la seule différence résidait dans l'emploi de l'opérateur d'adaptation $(\mathtt{r}_1, \mathcal{A}f_{\mathtt{r}_1})$. Cet opérateur peut être découvert en adaptant $(\mathtt{r}_1, \mathcal{A}_{\mathtt{r}_1})$ à la première forme. Dans le cas d'adaptation triviales de méthodes d'adaptation, un utilisateur relativement autonome peut produire lui-même une solution sans avoir à attendre d'aide ni des concepteurs ni des ingénieurs de la connaissance. De plus, il peut librement diffuser sa nouvelle solution en partageant son nouvel opérateur d'adaptation avec les autres utilisateurs et/ou les concepteurs. Cette fonctionnalité confère au système des propriétés proches de celles utilisées dans le système DIAL [13].

Cette possibilité a d'autres avantages par rapport aux systèmes de RÀPC classiques. En revanche, le système en l'état actuel est dépendant des choix de l'utilisateur et n'a pas appris à résoudre de lui-même les problématiques de changement de contexte. En effet, à chaque fois qu'un problème peut avoir plusieurs solutions, l'utilisateur doit lever l'ambiguïté en renseignant le contexte. Nous allons donc montrer comment les traces peuvent permettre au système d'apprendre interactivement à prendre en compte le contexte. La figure 4 représente une partie d'une `M-trace` correspondant à une activité dans *FLATLAND*. Les épisodes (nos cas) ont été découverts en utilisant des transformations de traces et une analyse de similarité. Cette `M-trace` contient non seulement des cas mais aussi d'autres observations (cercles, losanges et étoiles). Notre hypothèse est que les connaissances relatives au contexte peuvent être trouvées par l'utilisateur grâce à de simples interactions avec la `M-trace`. Dans cet exemple, imaginons que l'étoile symbolise une déclaration de guerre. Pour l'instant, l'observation de cet élément n'a pas été prise en compte dans le raisonnement. Toutefois, l'utilisateur peut très bien indiquer au système que cette information est importante pour résoudre automatiquement le problème de l'ambivalence du contexte. Il lui suffit pour cela d'expliquer une fois au système le rôle de cette observation et le système peut alors l'apprendre instantanément.

Nous pouvons également considérer d'une autre façon la dernière adaptation de l'épisode : comme une trace est temporellement ordonnée, la logique temporelle [3] peut être utilisée pour identifier le contexte. Dans l'exemple, l'épisode 3 a été adapté comme s'il s'agissait d'un combat. Comme l'épisode 4 suit, l'utilisateur peut spécifier une règle de contexte temporelle telle que : *si l'épisode d'adaptation précédent contient $\mathcal{A}f_{\mathtt{r}_1}$, alors utiliser les opérateurs d'adaptation du mode "combat"*.

Cet exemple simple montre quels genres d'améliorations peuvent être réalisées en utilisant les traces comme support pour trouver des cas. La limitation "*trop bien structuré et donc trop contraint*" peut être évitée en définissant de nouvelles connaissances sur le contexte. La trace offre un support concret à l'utilisateur et lui donne la possibilité d'expliquer directement au système ce dont il a besoin pour résoudre des ambiguïtés et surtout, comment le faire. L'acquisition de connaissances a des répercussions immédiates sur le processus de raisonnement qui peut alors évoluer dynamiquement. Avec une telle approche, la limitation des "*connaissances figées qui évoluent*" n'est plus d'actualité et l'expérience de l'utilisateur peut être captée facilement.

De nombreux domaines dans lesquels le contexte joue un rôle important pourraient bénéficier de cette approche interactive du RÀPC. Le principal problème dans ce champ de recherche est la multiplicité des points de vues liée à la multiplicité des utilisateurs. Le RÀPET dépasse l'hypothèse "*c'est bon pour moi maintenant donc ce sera toujours bon pour tout le monde*" et peut être utilisé pour s'adapter à des comportements hétérogènes. Il offre la possibilité au RÀPC de résoudre des problèmes moins structurés, et donc de rester compétitif et efficace sur des domaines à forte plasticité. Nous proposons d'ailleurs une étude plus complète sur un exemple concret dans [8] et détaillons ces aspects dans la section suivante.

5 Questions de recherche

Afin de construire un système de RÀPC basé sur les traces, plusieurs problèmes doivent être résolus. Ces problèmes sont décrits brièvement ci-dessous et leur étude constitue une de nos perspectives de recherche pour les années à venir.

Des cas aux traces, remémoration et adaptation : dans une approche à base de traces, les cas ne sont ni stockés ni organisés dans une structure figée. Ils sont seulement présents implicitement dans la trace. Ainsi, avant même de pouvoir faire du RÀPC, il est nécessaire de construire les cas à partir de la trace. Ceci souligne l'importance de l'élaboration dans le RÀPET. Par ailleurs, grâce aux traces, l'utilisateur peut voir directement l'évolution du traitement de sa requête. Par exemple, la réflexivité du processus de traçage donne à l'utilisateur la possibilité d'identifier immédiatement une mesure de similarité inadéquate : il peut alors décider d'interrompre le processus à cet instant précis. Dans ce cas, le système de RÀPET devrait lui permettre de choisir entre raffiner la mesure, en utiliser une autre, ou encore continuer le cycle en utilisant d'autres sources de connaissances (connaissances d'adaptation ou connaissances sur le contexte). L'automatisation du processus d'adaptation soulève également des problèmes plus compliqués : même s'il est possible d'effectuer des adaptations en contexte, il est nécessaire de mémoriser ces connaissances d'adaptation et de contexte dans des endroits spécifiques, en particulier si l'on souhaite les partager entre plusieurs utilisateurs. Les différents types de connaissances sont réellement liés dans le RÀPET et il est désormais facile de naviguer d'un type de connaissance à l'autre.

Extension de contexte "horizontal" et "vertical" : ce processus doit être amélioré afin de pouvoir s'affranchir de la limitation du cas "*trop bien structuré et donc trop contraint*". Lorsqu'un problème ne peut être résolu avec un épisode retrouvé, cet épisode doit être étendu (par exemple, en regardant ce qui s'est passé "avant" et "pendant" ou encore en utilisant des relations temporelles de Allen [3]). Mais l'extension de contexte ne doit pas se limiter à une analyse horizontale (et donc temporelle). La comparaison et l'adaptation d'épisodes est rendue possible grâce aux transformations des traces. Afin d'être comparées, les traces doivent parfois être généralisées, ce qui amène à construire plusieurs niveaux d'abstraction. Cette organisation verticale correspond à la prise en compte de l'histoire et de la provenance de la trace [11] : la navigation entre les niveaux de transformation (une approche *top-down*) peut aider l'utilisateur à formuler ce contexte en lui montrant ce qui pourrait se passer directement dans la `M-trace`. Ensuite, il peut indiquer au système quelles transformations sont incorrectes (un élément observé aurait pu être fusionné avec un autre ou simplement omis lors de la transformation). Cette approche devient alors *bottom-up*.

Vers un assistant pour le RÀPC : les compétences de l'utilisateur et son implication dans la démarche jouent un rôle important dans ce processus. Toute la problématique est de fournir les mécanismes nécessaires pour aider l'utilisateur dans ces opérations et pour lui permettre de déterminer quand le contexte doit être étendu. Toutefois, de telles aides sont dépendantes des domaines d'application et ne peuvent donc être implantées au niveau du TBMS. Il est préférable de se concentrer sur la conception d'assistants de RÀPET. Un grand défi serait de parvenir à la spécification d'interfaces génériques entre le TBMS, les assistants et les visualisateurs. Une tâche classique, que devraient proposer tous les systèmes de ce type, serait de leur permettre de rejouer des épisodes ou des solutions retrouvés. Ces systèmes devraient également fournir des outils pour identifier ce qui est

pertinent ou non et pour faciliter l'appropriation du système par les utilisateurs, en prenant en compte notamment la variété de leurs modalités d'interactions. Les assistants seraient également chargés d'aider au partage des traces entre les utilisateurs. Enfin, ils devraient être considérés comme des opérateurs capables de guider les TBMS de façon générique.

6 Travaux connexes et applications

Il existe de nombreux travaux exploitant les traces informatiques pour faire du diagnostic de comportement humain [9] ou de comportement de logiciels complexes [16]. À notre connaissance, si certains de ces travaux font de la fouille de motifs séquentiels [15], aucun n'exploite ces traces à des fins de réutilisation par adaptation de contexte. Réciproquement, le RÀPC a étudié les problèmes de multi-points de vue [10] et a développé également la notion de RÀPC conversationnel [2]. Le RÀPC conversationnel donne à la phase d'élaboration une grande importance : elle permet d'une part de décrire progressivement les valeurs d'un cas cible en utilisant les connaissances du système pour compléter les descriptions et guider l'utilisateur ; d'autre part, elle permet aussi de concevoir de nouvelles façons de décrire des cas. Toutefois, le système n'utilise pas l'aide de traces d'expérience permettant d'étendre le contexte d'un cas et de faciliter le raffinement et la construction de nouveaux cas, bien que cela puisse être utile pour étendre facilement le contexte d'étude par exemple.

Le RÀPET pourrait être efficacement utilisé dans des types d'applications variés : outils de résolution de problèmes, aides techniques pour le handicap, systèmes d'aide à l'apprentissage ou à l'enseignement, etc. Les domaines d'application sont également nombreux et nous pensons que des résultats prometteurs pourraient être obtenus avec des applications du web [8] et dans les outils de réseaux sociaux comme ceux décrits dans [5].

7 Discussion

Dans cet article, nous avons proposé une nouvelle approche du RÀPC qui facilite la réutilisation de l'expérience en contexte et qui est davantage centrée utilisateur. Cette approche repose sur l'utilisation des traces d'interactions. Elle résulte d'une synergie entre plusieurs travaux de recherche conduits au sein de l'équipe SILEX [2] du LIRIS. En effet, les différentes thématiques de recherche de l'équipe (RÀPC, traces, acquisition de connaissances, ingénierie de la connaissances, outils et techniques d'assistance, ergonomie, etc.) gravitent toutes autour d'une question unique : *comment raisonner et apprendre à partir de l'expérience ?*

L'approche que nous proposons s'appuie sur une théorie de la trace maintenant robuste ainsi que sur un ensemble d'outils permettant de gérer les traces et de les utiliser comme conteneurs de connaissances pour le RÀPC. Les traces permettent alors de repousser certaines limites des systèmes actuels de RÀPC comme cela a été décrit plus haut :

- sur la question du cas **"trop bien structuré et donc trop contraint"** : comme les cas sont dynamiquement élaborés à partir des traces, ils sont toujours placés dans leur contexte. Grâce à cela, nous nous affranchissons de la limitation liée à la structure rigide des cas qui ne nous permettait pas, jusqu'à maintenant, d'avoir une représentation de l'expérience fiable et surtout évolutive. L'élaboration dynamique des cas est possible grâce aux traces transformées et aux signatures expliquées de cas. C'est grâce au TBMS que les traces reformulées sont produites. En s'appuyant sur le TBMS, chaque utilisateur peut transformer des traces en utilisant ses propres connaissances et en fonction du problème qu'il doit résoudre. Avec cette approche, chaque utilisateur peut avoir son propre point de vue sur le problème.
- sur la question des **"connaissances figées qui évoluent"** : cette approche interactive du RÀPC permet d'effectuer une révision opportuniste des connaissances de similarité et d'adaptation. Un échec de remémoration ou d'adaptation est souvent lié à un problème dans les connaissances

[2] SILEX, Supporting Interactions and Learning by EXperience, http ://liris.cnrs.fr/silex

disponibles. Une fouille interactive de la trace permet alors d'élaborer, en collaboration avec l'utilisateur, les connaissances nécessaires à la réparation de l'échec. Les fonctionnalités de fouille sont fournies par le TBMS.

– sur la question du **"c'est bon pour moi maintenant donc ce sera toujours bon pour tout le monde"** : Un utilisateur expérimenté peut avoir envie de réutiliser son expérience dans le système. Pour cela, il doit être capable de concevoir de nouvelles signatures expliquées de cas. Afin de faciliter sa tâche, le système doit lui fournir des mécanismes de fouille de traces personnalisables en utilisant un ensemble de contraintes sur le problème [15]. Ainsi, les signatures expliquées de cas (et leurs mesures de similarité associées), sont co-construites par l'utilisateur et par le système.

L'association du RÀPC et des traces permet de redonner à l'utilisateur du système un rôle central. Les utilisateurs sont impliqués dans l'élaboration d'un problème, dans la remémoration d'une expérience passée et dans l'adaptation d'une solution. Ainsi, ils fournissent au système une source continue pour l'acquisition de connaissances en leur donnant la possibilité d'élaborer des connaissances au cours de chacune des interactions. L'utilisation des traces permet d'envisager d'une nouvelle façon l'ingénierie des systèmes de RÀPC. Les utilisateurs "re-conçoivent" en permanence les applications en fonction de leurs besoins et de leurs connaissances.

L'utilisation des traces soulève un certain nombre de questions relatives aux problématiques des interactions homme-machine. Les traces étant réflexives, leur visualisation nous est relativement facile. Nous savons ce que nous avons fait et nous sommes capables de comprendre le sens de nos actions. Cependant, utiliser les traces afin d'apprendre à partir de l'expérience est une tâche plus complexe. Elle se complexifie encore si nous ne limitions pas le concept d'expérience à l'expérience d'un individu unique mais mais si nous l'étendons au concept d'expérience partagée. En effet, une trace peut être le résultat de plusieurs interactions entre plusieurs utilisateurs. D'une manière similaire, un utilisateur peut avoir envie de bénéficier de traces provenant d'autres utilisateurs. Dans ce contexte, nous soulignons l'importance du concept de négociation de sens qui a été introduit et développé par Stuber dans [18]. De plus, apprendre à partir des traces nécessite davantage d'interfaces fournissant des outils permettant la transformation de traces et l'interaction avec celles-ci. Concevoir des interfaces efficaces pour supporter l'apprentissage à partir de l'expérience est une garantie de succès pour les systèmes de RÀPC dotés de facultés d'apprentissage continu.

L'importance que prennent les logiciels dans nos vies quotidiennes augmente de pair avec celle des interactions homme-machine. Ainsi, la capacité à faciliter les interactions dans le but de réutiliser efficacement l'expérience est un enjeu majeur pour les systèmes futurs. Les traces permettent d'envisager davantage de modalités d'interactions dans les systèmes et particulièrement de combiner plusieurs modalités d'interactions dans un système unique. Ainsi, plusieurs utilisateurs avec différentes compétences, capacités ou habitudes seront capables de partager leurs expériences. Nous envisageons d'expérimenter cette approche dans deux domaines d'application principaux : les environnements informatiques pour l'apprentissage humain et les aides techniques pour le handicap. Toutefois, de nombreux autres domaines d'application existent et mériteraient d'être étudiés.

Nous pensons que le raisonnement à partir de traces aura un impact significatif dans les applications de partage de l'expérience et en particulier dans les applications basées sur le web. Nous espérons que, grâce aux traces d'interactions, les générations futures de systèmes de RÀPC seront de véritables systèmes de raisonnement basés sur le partage de l'expérience, utilisés et partagés par des communautés d'utilisateurs hétérogènes.

Références

[1] Edwin A. Abbott. *Flatland, a romance of many dimensions*. E.Books, Second revised edition, 1884.

[2] David W. Aha, Leonard A. Breslow, et Hector Munoz-Avila. Conversational CBR. *Applied Intelligence*, 2001.

[3] JF Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2) :123–154, 1984.

[4] Ralph Bergmann, Sean Breen, Mehmet Göker, Michel Manago, Sascha Schmitt, Jörgen Schumacher, Armin Stahl, Emmanuelle Tartarin, Stefan Wess, et Wolfgang Wilke. The inreca-ii methodology for building and maintaining cbr applications. In *Proceedings of the 6th German Workshop on CBR*, 1998.

[5] Peter Briggs et Barry Smyth. Provenance, trust, and sharing in peer-to-peer case-based Web search. In *Advances in CBR proceedings*, pages 89–103. Springer-verlag, 2008. 9th European Conference on CBR, Trier, Germany.

[6] Amélie Cordier, Béatrice Fuchs, Jean Lieber, et Alain Mille. Interactive Knowledge Acquisition in CBR. In *Proceedings of the workshop on Knowledge Discovery and Similarity (at ICCBR'07)*, pages 85–94. Workshop of the seventh International Conference on CBR Workshop, 2007.

[7] Amélie Cordier. *Interactive and Opportunistic Knowledge Acquisition in CBR*. PhD Thesis, Université Claude Bernard Lyon 1, November 2008.

[8] Amélie Cordier, Bruno Mascret, et Alain Mille. Extending CBR with Traces. In *IJCAI's Workshop on Grand Challenges in Reasoning from Experience*, 2009.

[9] V. Diekert et G. Rozenberg. *The Book of Traces*. World Scientific Publishing, Singapore, 1995.

[10] Nikos Karacapilidis, Brigitte Trousse, et Dimitris Papadias. Using CBR for argumentation with multiple viewpoints. In *CBR Research and Development (ICCBR'97), volume 1266 of Lecture Notes in AI*, pages 541–552. Springer, 1997.

[11] David Leake et Scott A. Dial. Using case provenance to propagate feedback to cases and adaptations. In *Advances in CBR proceedings*, pages 89–103. Springer-verlag, 2008. 9th European Conference on CBR, Trier, Germany.

[12] David B. Leake, Andrew Kinley, et David Wilson. *CBR : Experiences, Lessons and Future Directions*, chapter Learning to improve Case Adaptation by Introspective Reasoning and CBR, pages 185–196. AAAI Press, MIT Press, 1996.

[13] David B. Leake, Andrew Kinley, et David Wilson. Linking adaptation and similarity learning. In *Eighteenth Annual Conference of the Cognitive Science Society*, 1996.

[14] Bruno Mascret. Apprendre à assister l'utilisateur à partir de l'expérience tracée ? Master's thesis, Université Claude Bernard Lyon 1, September 2008.

[15] M.Gaber, A.Zaslavsky, et S. Krishnaswamy. Mining data streams : a review. *Sigmod Rec*, 2005.

[16] S. Penelope et C. Fisher. Exploratory sequential data analysis : foundations. *Human Computer Interaction*, 9(3) :251–317, 1994.

[17] Lotfi Settouti, Yannick Prié, Pierre-Antoine Champin, Jean-Charles Marty, et Alain Mille. A TBMS framework : Models, languages and semantics. Research Report, http ://hal.inria.fr, 2009.

[18] Arnaud Stuber. *Co-construction de sens par négociation pour la réutilisation en situation de l'expérience tracée* . PhD Thesis, Université Claude Bernard Lyon 1, 2007.

Traitement de l'hétérogénéité dans un système de RàPC basé sur une ontologie

Amjad Abou Assali[1], Dominique Lenne[1], Bruno Debray[2]
[1]Université de Technologie de Compiègne, CNRS
HEUDIASYC
{amjad.abou-assali,dominique.lenne}@utc.fr
[2]INERIS*
bruno.debray@ineris.fr

Résumé

Cet article présente notre plate-forme de raisonnement à partir de cas pour le diagnostic, appelée COBRA. Cette plate-forme est basée sur une structure ontologique intégrant les connaissances de domaine et les cas. Nous travaillons actuellement sur la défaillance de capteurs de gaz installés sur des sites industriels. Vu la nature de notre domaine d'application, les cas peuvent être décrits par un concept ou une instance quelconque de l'ontologie de domaine. Cela conduit ainsi à une base de cas hétérogène, ce qui complique la remémoration des cas. Nous présentons dans cet article notre approche pour pallier le problème de l'hétérogénéité en introduisant les notions de région de similarité et de rôle pour les attributs de cas. Enfin, nous montrons à travers un exemple l'intérêt de l'approche proposée.

Mots clés : Raisonnement à partir de cas, Ontologie, Cas hétérogènes, Région de similarité.

1 Introduction

La gestion des risques est une préoccupation importante sur les sites industriels concernés par la directive SEVESO vu la présence de substances et de processus dangereux *in situ*. Pour réduire les risques potentiels, des barrières de sécurité sont proposées par des experts. Le choix de ces barrières dépend du type du processus à contrôler et de la situation dangereuse ainsi que des conditions environnementales sur site. Toutefois, ces barrières peuvent échouer à assurer la fonction de sécurité pour laquelle elles ont été installées, et des accidents peuvent se produire. Dans ce contexte, un expert industriel intervient pour diagnostiquer le dysfonctionnement des barrières. Dans un premier temps, il essaie de se remémorer des expériences de dysfonctionnement observées dans des situations similaires. L'hypothèse que l'expert fait est que "si une barrière n'a pas bien fonctionné dans une autre situation similaire, il est fortement probable qu'elle ne fonctionne pas, dans la situation actuelle, *pour des raisons similaires*". La façon dont l'expert procède pour diagnostiquer nous a orienté vers une approche de raisonnement à partir de cas (RàPC) [16]. Le RàPC est une approche de résolution de problèmes ayant pour objectif de résoudre un nouveau problème, appelé *problème cible*, à l'aide d'un ensemble de problèmes déjà résolus, appelés *problèmes sources*. Quand l'expert avance dans son diagnostic, il cherche parfois à obtenir plus d'informations sur la situation actuelle pour pouvoir trouver la bonne cause de défaillance. Cela nous conduit à adopter une approche conversationnelle où les cas sont enrichis au fur et à mesure tant que le diagnostic proposé par le système ne satisfait pas l'expert.

Notre travail est actuellement appliqué au diagnostic de la défaillance de capteurs (détecteurs) de gaz, des dispositifs de sécurité jouant un rôle principal dans le fonctionnement des barrières de sécurité (vannes, extincteurs automatiques, *etc.*) installées sur des sites industriels. Ces capteurs ont pour fonction de déclencher une alarme lorsqu'il y a une fuite de certains gaz quelque part dans le site. Une action de sécurité sera alors mise en place automatiquement ou par une intervention humaine. Dans ce contexte, un cas représente un diagnostic

*Institut National de l'Environnement industriel et des RISques.

de la défaillance d'un capteur de gaz dans un environnement donné ; *i.e.* c'est la description du contexte du diagnostic ainsi que la (les) cause(s) de la défaillance trouvée(s).

Nous présentons, dans cet article, COBRA (Conversational Ontology-based CBR for Risk Analysis), une plate-forme de RàPC basée sur une ontologie. Cette plate-forme est basée sur des modèles de connaissances représentés par des ontologies pour décrire le domaine et les cas. Vu la nature de notre domaine d'application, COBRA permet à l'utilisateur de décrire ses cas par un concept ou une instance quelconque de l'ontologie de domaine. Cela conduit donc à une base de cas hétérogène où les cas peuvent être décrits par différents attributs, ce qui complique la remémoration des cas. Nous présentons notre approche pour résoudre les problèmes de l'hétérogénéité, et nous montrons à travers un exemple l'intérêt de l'approche proposée.

2 État de l'art

Les approches *knowledge-intensive* du RàPC (KI-CBR) sont celles pour lesquelles les connaissances du domaine jouent un rôle fondamental (et pas uniquement la base de cas). L'intégration des connaissances génériques du domaine d'application dans les systèmes de KI-CBR a été un aspect important dans plusieurs projets. Dans l'architecture de CREEK [1], nous trouvons un couplage assez fort entre les connaissances des cas et celles du domaine. Ainsi, les cas sont immergés dans un modèle générique du domaine représenté par un réseau sémantique. Fuchs et Mille ont proposé une modélisation du RàPC au niveau connaissance [10]. Ils ont distingué quatre modèles de connaissance : 1) le modèle conceptuel du domaine décrivant les concepts utilisés pour décrire l'ontologie du domaine indépendamment du raisonnement ; 2) le modèle de cas qui sépare le cas en problème, solution, et trace de raisonnement ; 3) les modèles de tâches de raisonnement qui comprennent un modèle de spécification et un autre de décomposition de tâches ; 4) et les modèles supports du raisonnement. D'Aquin *et al.* ont travaillé sur l'intégration du RàPC dans le Web sémantique [6]. Pour cela, ils ont proposé une extension de OWL (Ontology Web Language) permettant de représenter les connaissances d'adaptation du RàPC. L'expression des connaissances du domaine et des cas en OWL leur a permis de rajouter au système de RàPC les capacités de raisonnement propres à OWL en exploitant, par exemple, la subsomption et l'instanciation. Dìaz-Agudo et Gonzàlez-Calero ont proposé une architecture indépendante du domaine qui aide à l'intégration d'ontologies dans les applications de RàPC [7]. Leur approche consiste à construire des systèmes intégrés qui combinent des connaissances spécifiques aux cas avec des modèles génériques des connaissances du domaine. Ils ont présenté CBROnto [8], une ontologie de tâche/méthode qui fournit le vocabulaire nécessaire pour décrire les éléments impliqués dans les processus de RàPC, et qui permet également d'intégrer différentes ontologies de domaine. CBROnto a été réutilisée plus tard par jCOLIBRI, un framework orienté-objet (en JAVA) assez puissant pour la construction de systèmes de RàPC [13, 9]. jCOLIBRI sépare la gestion des bases de cas en deux aspects : la persistance et l'organisation en mémoire, ce qui permet d'avoir différents supports de stockage de cas (fichiers textes/XML, ontologie, *etc.*) accessibles via des connecteurs spécifiques. Toutefois, jCOLIBRI ne permet pas le traitement de bases de cas dynamiques et hétérogènes, ce qui nous a amené à développer une couche supplémentaire pour pallier ce manque.

3 Architecture de COBRA

Plusieurs architectures des systèmes de RàPC ont été proposées dans la littérature. Ces architectures partagent plus ou moins les mêmes composantes. Aamodt et Plaza [2] ont présenté le cycle fameux de RàPC constitué des processus : REMÉMORER, RÉUTILISER, RÉVISER, et MÉMORISER. Un cinquième processus, ÉLABORER, a été distingué plus tard pour représenter la phase d'élaboration d'un cas cible (voir [14]). Lamontagne et Lapalme [11] ont présenté une vue globale où l'on trouve les processus hors-ligne/en-ligne, et les connaissances (*knowledge containers*) permettant de préserver et exploiter les expériences passées. Comme le montre la figure 1, COBRA est inspiré par ces architectures et est composé de deux parties principales [4] :

– *Processus* : cette partie est constituée d'un processus hors-ligne, *authoring des cas*, et de six processus de raisonnement : ÉLABORER, REMÉMORER, DIAGNOSTIQUER, ENRICHIR, VALIDER et MÉMORISER. Dans les systèmes conversationnels de RàPC pour le diagnostic, il est important de distinguer deux processus fondamentaux, *diagnostiquer* et *enrichir*. Dans la phase de diagnostic, le système essaie d'identi-

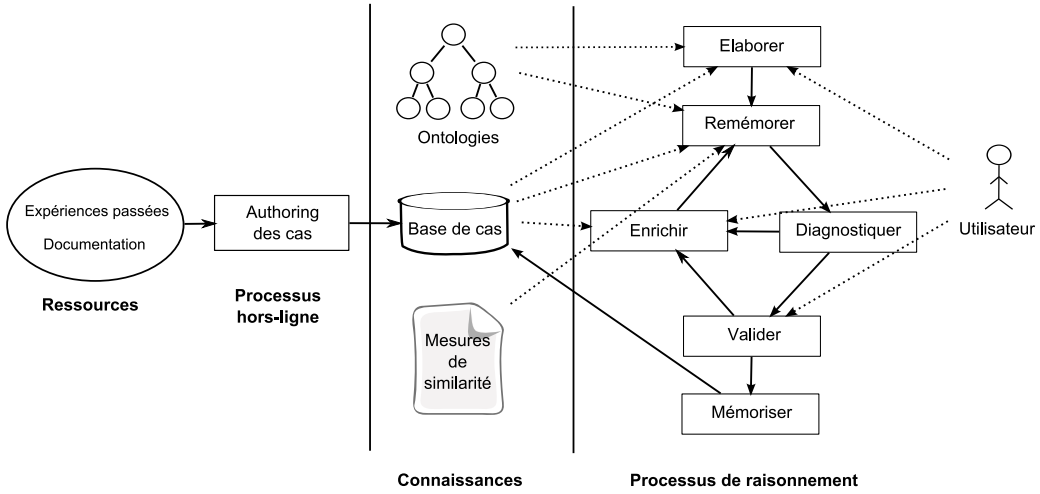


FIG. 1 – Architecture de COBRA.

fier les causes de défaillance à partir des cas similaires. Si aucune cause n'est trouvée, ou si le diagnostic proposé par le système n'a pas été validé par l'utilisateur, le système demande à l'utilisateur d'enrichir la description du cas cible pour chercher de nouveaux cas similaires. Ce cycle se reproduit jusqu'à ce qu'un bon diagnostic soit proposé par le système, ou qu'aucune solution ne puisse plus être trouvée.

– *Connaissances* : nous distinguons trois types de connaissances ("knowledge containers") : vocabulaire, base de cas, mesures de similarité. Par contre, nous n'avons pas de règles d'adaptation pour le moment. Dans les systèmes de KI-CBR, les ontologies jouent un rôle important pour représenter ces connaissances. Elles peuvent représenter le vocabulaire pour décrire les cas, ou bien la structure de connaissances où les cas sont localisés. Elles peuvent également intervenir pour permettre un raisonnement sémantique dans les méthodes de calcul de similarité.

Pour décrire le vocabulaire et la base de cas, notre système se base sur deux modèles de connaissances : les modèles de domaine et de cas.

3.1 Modèle de domaine

Ce modèle représente les connaissances autour des barrières de sécurité, notre domaine d'application, sous forme d'ontologies. Pour clarifier les types d'ontologies développées, nous avons suivi la classification proposée par Oberle [12]. Tout d'abord, notre objectif était de développer une ontologie réutilisable dans plusieurs domaines de la sécurité industrielle. Nous avons donc développé une ontologie *noyau* qui contient des concepts génériques sur la sécurité industrielle tels que : ÉVÉNEMENT, PHÉNOMÈNE DANGEREUX, ÉQUIPEMENT, SUBSTANCE, *etc.* Ensuite, nous avons développé une ontologie *de domaine* dont les concepts sont des spécialisations de l'ontologie noyau (voir FIG. 2). Cette ontologie décrit le domaine des barrières de sécurité, en particulier les capteurs de gaz. Un capteur de gaz est conçu pour fonctionner dans certaines *conditions ambiantes* (humidité, température, présence de poussière, *etc.*) afin d'assurer la *fonction de sécurité* consistant à détecter la présence de certains gaz dangereux *in situ*. Pour cela, nous trouvons dans cette ontologie des concepts tels que : CAPTEUR DE GAZ, FONCTION DE SÉCURITÉ, CONDITION AMBIANTE, *etc.* Les ontologies sont décrites en OWL Lite, et elles ont été développées par plusieurs experts de l'INERIS avec l'aide d'un expert en ontologie [3].

3.2 Modèle de cas

Un cas dans notre système est un cas de diagnostic. Il contient trois parties principales : la partie *description*, qui décrit le contexte dans lequel a été réalisé le diagnostic, la partie *mode de défaillance*, et la partie *causes*. Prenons, par exemple, la défaillance de capteurs de gaz. La partie description peut contenir : le gaz à mesurer, la technologie du capteur utilisé, le seuil de la concentration d'alarme (*i.e.* à quelle concentration du gaz le capteur

doit déclencher une alarme), *etc.* Le mode de défaillance peut être :

- une fausse alarme, par exemple : alarme intempestive, concentration surestimée, erreur sur la nature du gaz ;
- une absence de détection de gaz, par exemple : échec de fonctionnement, concentration sous-estimée ;
- une détection tardive quand le capteur a un temps de réponse élevé ;
- une défaillance dans la transmission des informations.

Enfin, la cause de défaillance peut être :

- technologie du capteur inadaptée au gaz à mesurer ;
- capteur contaminé (*e.g.* un gaz présent sur le site empêche le capteur de bien fonctionner) ;
- mauvais calibrage du capteur ;
- mauvaise configuration de l'alarme ;
- capteur insuffisamment maintenu.

Un cas est généralement décrit par un couple (*problème*, *solution*). Selon notre modèle, les parties *description* et *mode de défaillance* correspondent à la partie *problème*, et la partie *causes* correspond à la partie *solution*.

Pour améliorer la communication entre la base de cas et le modèle de domaine, notre modèle de cas est représenté à l'aide d'une ontologie qui intègre le modèle de domaine. Nous nous inspirons en cela de l'approche utilisée dans jCOLIBRI. Cette ontologie contient les concepts racines suivants (FIG. 2) :

- CBR-CASE qui subsume les concepts représentant les différents types de cas qui peuvent exister dans le système.
- CBR-DESCRIPTION qui subsume les concepts représentant les parties d'un cas (mode de défaillance, cause, *etc.*).
- CBR-INDEX qui permet d'intégrer les concepts du modèle de domaine.

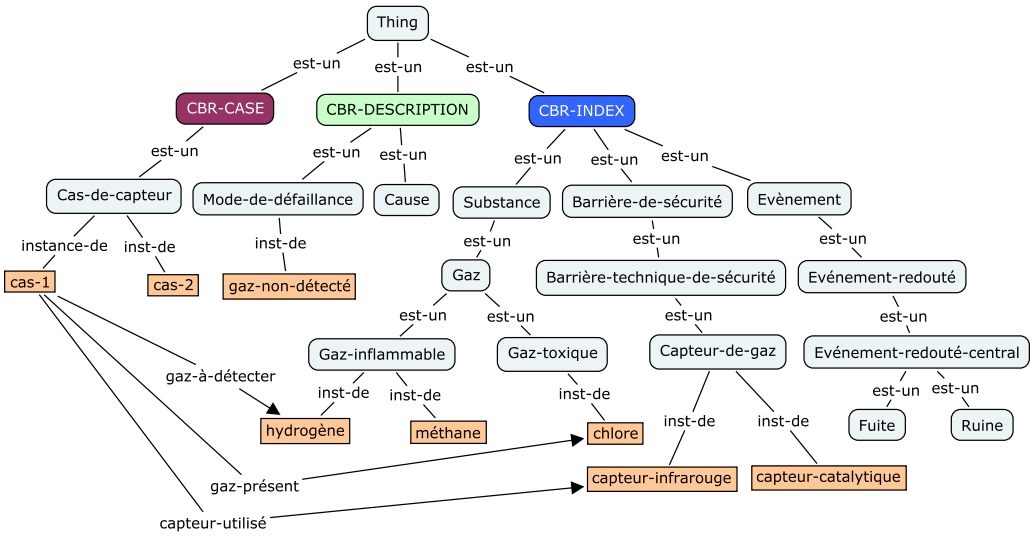


FIG. 2 – Modèle de cas.

Les cas sont alors représentés par des instances de l'ontologie, et ils ont donc deux types d'attributs :

- des attributs simples correspondant à des propriétés *data-type* de l'ontologie qui prennent des valeurs simples, *i.e.* string, int, float, *etc.*
- des attributs complexes correspondant à des instances (ou des concepts) de l'ontologie.

4 Processus de RàPC

Nous décrivons dans la suite les premiers processus de notre cycle de RàPC : *authoring* des cas, élaboration d'un cas cible (d'une requête) et remémoration de cas.

4.1 Authoring des cas

Pour initialiser les bases de cas, différents moyens peuvent être envisagés [17]. Dans notre travail, des experts du domaine ont été sollicités, dans un premier temps, pour construire des cas de diagnostic de la défaillance de capteurs de gaz à partir de leur expérience. Puis, des cas ont été rajoutés à partir des documentations existantes. Les cas sont décrits par des concepts du modèle de domaine. Toutefois, nous nous sommes rendu compte que nos cas ne partagent pas toujours la même structure, autrement dit, les mêmes attributs. Pour cela, deux objectifs ont émergé :

- Tout d'abord, les experts ne doivent pas être limités à un certain nombre d'attributs pour décrire leur expérience. Ils doivent pouvoir utiliser tout concept du modèle de domaine dont ils ont besoin. Cela conduit donc à une base de cas hétérogène, ce qui complique la phase de remémoration de cas.
- Comme les cas peuvent être décrits par différents attributs, il est utile d'aider l'expert durant l'authoring de son expérience. Pour ce faire, le système lui montre une liste des concepts les plus utilisés dans les cas similaires ordonnancés suivant leur importance. La base de cas est alors exploitée non seulement dans la phase de remémoration, mais aussi dans la phase d'authoring des cas.

Le processus d'élaboration de cas cibles (requêtes) est similaire. Par contre, des poids peuvent être associés aux attributs d'un cas cible pour être pris en compte dans la phase de remémoration (voir la section suivante).

4.2 Remémoration de cas

Dans cette phase, des mesures de similarité sont utilisées pour récupérer les cas similaires à un cas cible (requête). En général, avec les structures orientées-objet des cas, les mesures de similarité suivent le principe "local-global" [5, 15] qui dit : "le but est de déterminer la similarité entre deux objets, un objet représentant le cas source (ou une partie du cas) et un autre objet représentant le cas cible (ou une partie du cas). Cette similarité est appelée la similarité globale, et est calculée d'une manière récursive ; *i.e.* pour chaque attribut simple, une mesure de similarité *locale* détermine la similarité entre les deux valeurs d'attribut. En revanche, pour chaque attribut complexe, une mesure de similarité *globale* est utilisée. Enfin, les valeurs des similarités locales et globales sont agrégées, d'une manière récursive, pour donner la similarité globale des deux objets comparés".

D'un point de vue ontologique, le calcul de similarité entre deux concepts de l'ontologie peut être divisé en deux composantes [5, 13] : *une similarité basée-concept* (ou similarité intra-classe) qui dépend de l'emplacement des concepts dans l'ontologie, et *une similarité basée-slot* (ou similarité inter-classe) qui dépend des valeurs des attributs communs des objets comparés.

4.2.1 Mesures de similarité

Les attributs d'un cas cible n'ont pas toujours la même importance dans le calcul de similarité. Ainsi, il est important de permettre à l'utilisateur d'associer à chaque attribut un certain poids. Dans notre travail, les poids peuvent être attribués à deux niveaux différents du cas cible :

- Les attributs simples peuvent avoir l'un des trois modes de calcul de similarité :
  - IGNORE : l'attribut n'a pas d'importance.
  - EXACT : qui permet de vérifier l'égalité stricte des valeurs de l'attribut.
  - NUMÉRIQUE : qui est applicable aux attributs numériques seulement. Plus les deux valeurs sont proches l'une de l'autre, plus elles sont similaires.
- Les attributs complexes ont un poids dans l'intervalle [0, 1]. En outre, chaque attribut simple d'un attribut complexe peut avoir l'un des trois modes de calcul (IGNORE, EXACT, NUMÉRIQUE).

Nous expliquons dans la suite les mesures de similarité utilisées dans COBRA.

Soit $Q = \{q_i : 1 \leqslant i \leqslant n, n \in \mathbb{N}^*\}$ un cas cible pour lequel on cherche des cas similaires, où q_i est un attribut simple ou bien complexe, et soit $\Omega = \{C_j : 1 \leqslant j \leqslant k, k \in \mathbb{N}^*\}$ la base de cas, où $C_j = \{c_{jl} : 1 \leqslant l \leqslant m_j, m_j \in \mathbb{N}^*\}$. Pour chaque attribut complexe, $q \in Q$ et $c \in C$, la similarité basée-concept, sim_{cpt}, est définie comme suit :

$$sim_{cpt}(q,c) = w_q * \frac{2 * prof(LCS(q,c))}{prof(q) + prof(c)} \quad (1)$$

où w_q est le poids associé à q, $prof$ est la profondeur d'un concept (ou d'une instance) dans l'ontologie, et LCS est le plus petit subsumant commun (*Least Common Subsumer*) de deux instances. Dans un cas particulier, quand q et c représentent la même instance, nous avons : $prof(LCS(q,c)) = prof(q)$.

La similarité basée-slot, sim_{slt}, est définie comme suit :

$$sim_{slt}(q,c) = \frac{\sum_{s \in CS} sim(q.s, c.s)}{|CS|} \quad (2)$$

où CS est l'ensemble des attributs simples en commun entre q et c (Common Slots), $|CS|$ est sa cardinalité, $q.s$ (ou $c.s$) représente l'attribut simple s de q (ou de c), et $sim(q.s, c.s)$ est la similarité entre ces deux attributs. Pour le moment, nous considérons seulement les deux premiers modes (IGNORE, EXACT), et donc $sim(q.s, c.s)$ est définie comme suit :

$$sim(q.s, c.s) = \begin{cases} 1 & \text{si} \quad (w_{q.s} = exact) \wedge (v_{q.s} = v_{c.s}) \\ 0 & \text{sinon} \end{cases}$$

où $w_{q.s}$ est le mode associé à l'attribut $q.s$, et $v_{q.s}$ est la valeur de cet attribut dans q.

La mesure globale de similarité entre les deux attributs complexes, q et c, est définie par la formule suivante [18] :

$$sim(q,c) = (1-\alpha) * sim_{cpt}(q,c) + \alpha * sim_{slt}(q,c) \quad (3)$$

où α est un paramètre d'expérience (actuellement, $\alpha = 0.4$).

Pour calculer la similarité basée-concept, chaque attribut complexe du cas cible est comparé à son attribut correspondant d'un autre cas source. Dans les bases de cas *homogènes*, tous les cas partagent la même structure prédéfinie, et donc la correspondance entre les attributs complexes des cas est déjà définie. En revanche, dans les bases de cas *hétérogènes*, cette correspondance n'est pas préalablement définie. En conséquence, avant de calculer la similarité basée-concept, il faut déterminer les attributs complexes correspondants.

Pour chaque attribut complexe $q' \in Q$, soit $c' \in C_j$ l'attribut complexe correspondant dans le cas $C_j \in \Omega$. Afin de déterminer c', nous avons considéré au départ que c'est l'attribut avec lequel q' a une similarité maximale, ce qui donne la formule suivante :

$$sim(q', c') = \max_{1 \leqslant l \leqslant m_j} (sim(q', c_{jl})) \quad (4)$$

Toutefois, nous avons constaté que cette définition n'est pas satisfaisante, et que d'autres conditions doivent être réunies. Le problème est qu'on arrive toujours à trouver c' suivant cette formule alors que q' peut n'avoir aucun attribut correspondant dans C_j. La première amélioration de cette formule serait de comparer la similarité $sim(q', c')$ obtenue dans C_j avec la similarité maximale obtenue sur l'ensemble des cas pour l'attribut q'. Cela conduit à la condition suivante :

$$\frac{sim(q', c')}{\max_{1 \leqslant j \leqslant k, 1 \leqslant l \leqslant m_j}(sim(q', c_{jl}))} \geqslant \beta \quad (5)$$

où β est un certain seuil que nous avons déterminé après une première validation du système (actuellement, $\beta = 0.6$).

La prise en compte de cette condition donne de meilleurs résultats, mais elle n'est pas toujours suffisante dans certains cas. Pour cela, nous proposons de prendre en compte les notions de région de similarité et de rôle d'attribut.

4.2.2 Régions de similarité

Nous considérons que les mesures de similarité ne sont pas toujours applicables sur tout couple de concepts (ou d'instances) de l'ontologie. Par exemple, un gaz ne devrait pas en principe être comparé à un équipement. Afin d'interdire de telles comparaisons, nous proposons de définir la notion de *région de similarité*. Une région de similarité est une sous-hiérarchie de l'ontologie où tous les concepts et les instances sont comparables entre eux (FIG. 3). La définition de ces régions se fait manuellement et dépend de l'application visée. Avant de calculer la similarité entre un attribut d'un cas cible et un autre d'un cas source, il faut vérifier si ces deux attributs appartiennent à la même région de similarité.

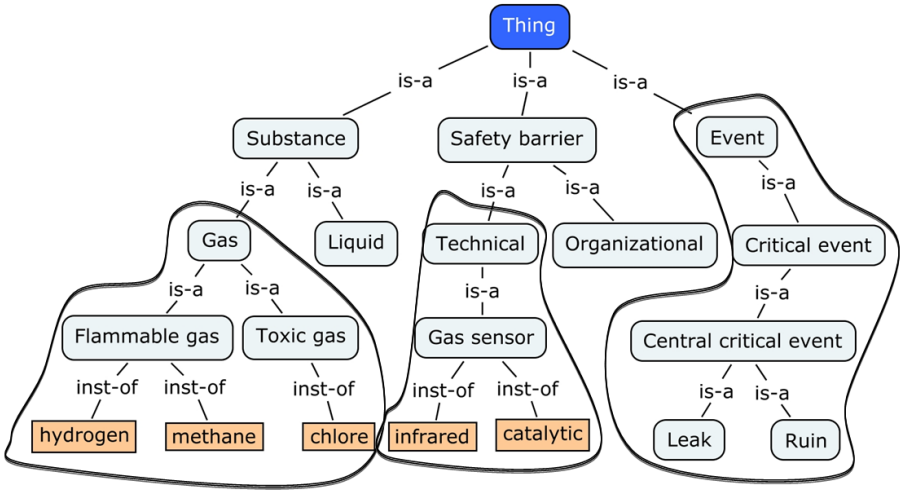


FIG. 3 – Exemple de régions de similarité.

4.2.3 Rôles d'attributs

Pour un attribut cible donné, plusieurs attributs sources peuvent parfois être trouvés. Prenons, par exemple, la partie suivante d'une description d'un cas : "Sur un site industriel, un capteur de gaz infrarouge a été utilisé pour **détecter** le méthane. D'autres gaz étaient **présents** sur le site dont l'hydrogène. Le capteur n'a pas bien fonctionné et une explosion s'est produite". Supposons maintenant une requête cherchant les cas où un capteur de gaz a été utilisé pour **détecter** l'hydrogène. En suivant l'approche présentée jusqu'à présent, nous trouvons que l'hydrogène du cas est l'attribut correspondant le mieux à l'hydrogène de la requête. Toutefois, ce n'est pas le résultat souhaité, car nous cherchons les cas où l'hydrogène était le gaz à détecter, et donc son attribut correspondant devrait être le méthane.

Pour lever cette ambiguïté, nous proposons de préciser le rôle de chaque attribut complexe pouvant conduire à des situations ambiguës. Ainsi, l'hydrogène du cas aurait pour rôle "**gaz présent**", et celui de la requête "**gaz à détecter**".

Afin de déterminer la correspondance entre les attributs complexes, nous identifions tout d'abord les attributs ayant le même rôle. Ensuite, nous appliquons les méthodes proposées (Formules 4, 5, régions de similarité) aux autres attributs.

5 Inférence de nouveaux attributs

L'ontologie de domaine nous a permis, jusqu'à présent, d'exploiter les relations de subsomption et d'instanciation dans les mesures de similarité proposées. Toutefois, l'ontologie peut jouer un rôle encore plus important dans le raisonnement. Pour un cas donné, il est parfois possible de déduire de nouveaux attributs à partir des attributs décrits explicitement. Prenons, par exemple, un cas où il y a un "dépoussiéreur humide" (un équipement). Suite à la présence de cet équipement, on doit pouvoir déduire la *présence de poussière* (une condition ambiante) dans l'environnement, ce qui n'est pas décrit explicitement dans le cas. Grâce à l'ontologie, nous

pouvons inférer ce genre d'information en raisonnant sur les relations (horizontales) existant entre les concepts (et entre les instances). Pour ce faire, nous avons intégré le moteur d'inférence JENA[1] qui se base sur un ensemble de règles d'inférence (FIG. 5). COBRA essaie ainsi d'inférer de nouveaux attributs de cas en utilisant JENA, ce qui donne lieu à des cas contenant des attributs explicites et d'autres inférés.

6 Plate-forme COBRA

Nous avons développé la plate-forme COBRA en tant qu'application JAVA basée sur Eclipse[2] grâce au framework RCP[3] (Rich Client Platform). Elle profite ainsi de beaucoup de fonctionnalités offertes par Eclipse. COBRA permet la construction de systèmes de RàPC dont les connaissances sont décrites par des ontologies. Nous travaillons actuellement sur la défaillance de capteurs de gaz installés sur des sites industriels. Dans ce contexte, COBRA est utilisé pour répondre à deux objectifs principaux : 1) Capitaliser les connaissances autour de la défaillance des capteurs de gaz ; 2) Fournir une aide aux experts et aux ingénieurs de sécurité pour pouvoir diagnostiquer les causes de défaillance des capteurs dans des conditions industrielles.

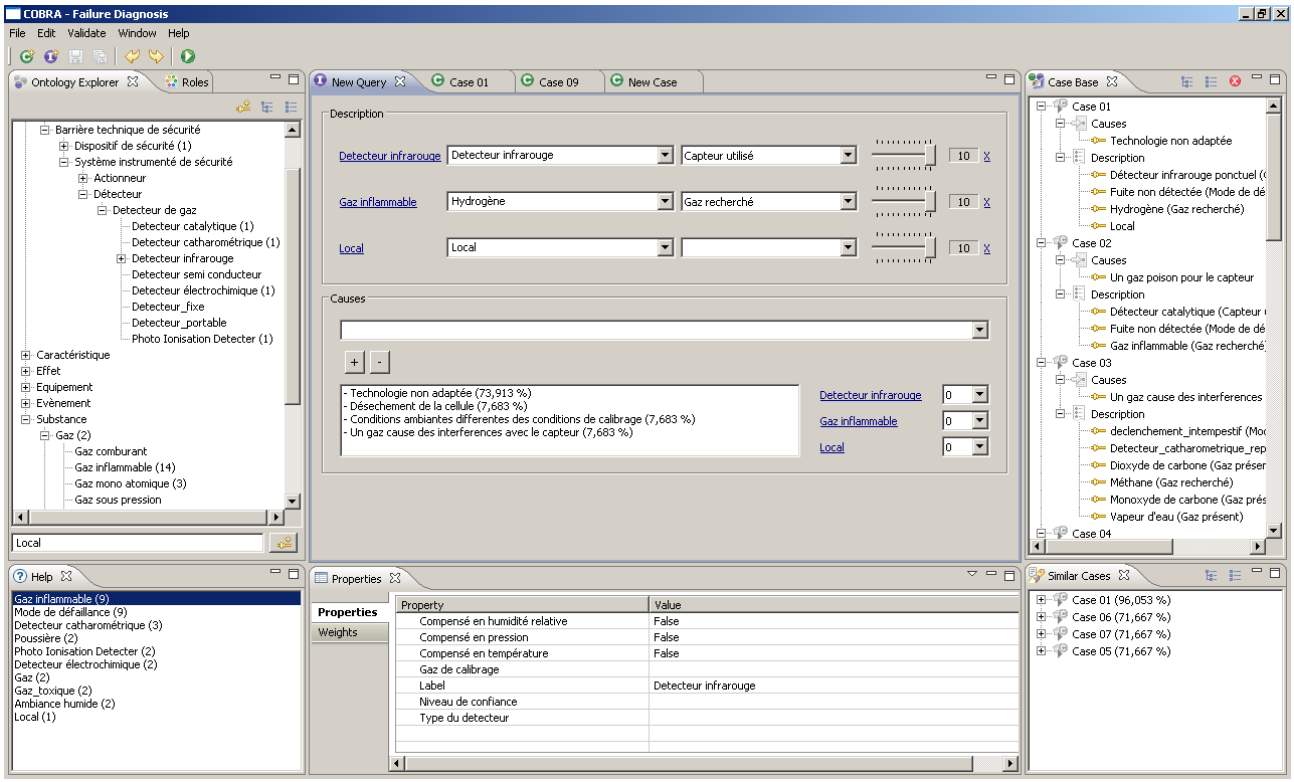


FIG. 4 – La plate-forme COBRA.

La plate-forme contient les onglets (viewers) principaux suivants (FIG. 4) : l'explorateur d'ontologie (en haut à gauche) qui montre les concepts de l'ontologie de domaine, la base de cas (en haut à droite), l'onglet des cas similaires par rapport à un cas cible, l'onglet d'aide qui propose à l'utilisateur les concepts candidats et les plus utilisés dans la base de cas, et l'onglet des propriétés permettant de modifier les valeurs de l'instance sélectionnée (d'un cas ou d'un attribut de cas).

L'authoring des cas se fait en créant un nouveau cas, puis en renseignant ses attributs simples et/ou en y ajoutant des attributs complexes (instances de l'ontologie). L'utilisateur peut choisir parmi les instances présentes, ou il peut créer et ajouter ses propres instances, ce qui permet d'enrichir la base de connaissances. Ensuite, l'utilisateur peut associer (ou créer) des rôles aux attributs complexes si nécessaire.

[1] http ://jena.sourceforge.net
[2] http ://www.eclipse.org
[3] http ://wiki.eclipse.org/index.php/Rich_Client_Platform8

COBRA permet aux cas d'avoir différentes structures, et donc de travailler avec une base de cas hétérogène. Pour développer COBRA, nous nous sommes appuyés sur l'API jCOLIBRI. Cette API ne permet pas le traitement de bases de cas dynamiques et hétérogènes, car les attributs de cas doivent être définis auparavant (dans le code source) avant de lancer la plate-forme. Pour cela, nous avons étendu cette API par une couche permettant de pallier ce manque (FIG. 5). Cette couche contient notre propre connecteur d'ontologie et le module de calcul de similarité ainsi que le fichier de règles d'inférence. Grâce à cette architecture, le développement d'un nouveau système de RàPC se fait en fournissant l'ontologie de domaine, et en paramétrant des fichiers XML de configuration.

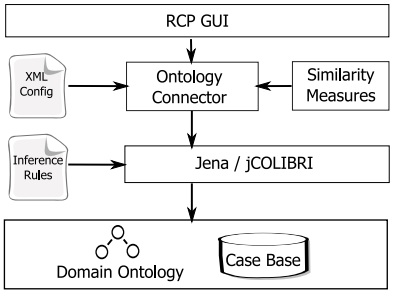


FIG. 5 – Les composantes de COBRA.

7 Résultats

Nous avons présenté dans cet article notre approche pour pallier le problème de l'hétérogénéité en se basant sur les calculs de similarité et en introduisant les notions de *région de similarité* et de *rôle d'attribut*. Pour pouvoir faire une première évaluation de l'approche proposée, des cas simples sur la défaillance de capteurs de gaz ont été rajoutés à la base de cas. Ces cas correspondent à des situations réelles observées *in situ* ou bien à des tests réalisés à l'INERIS durant des campagnes de qualification des capteurs. L'évaluation a été faite dans le but de montrer la qualité des cas similaires retournés par le système pour des requêtes données. L'évaluation nous a permis également de vérifier les solutions proposées ; *i.e.* la condition de la formule 5, les régions de similarité, et les rôles d'attributs. Nous expliquons dans la suite les résultats que nous obtenons à partir d'un exemple comportant quatre cas simples afin de montrer les améliorations apportées par chaque solution proposée.

En prenant les cas dans les tableaux suivants (TAB. 1), supposons une requête qui cherche les cas où un *détecteur PID* (détecteur PhotoIonisation) a été utilisé pour détecter des *vapeurs de solvant* (un gaz inflammable). Ce détecteur a été installé dans un environnement où il y avait de la *poussière* (condition ambiante).

Cas 1	
Attribut	Rôle
Détecteur PID	Capteur utilisé
Vapeur de solvant	Gaz recherché
Dépoussiéreur humide	Condition ambiante
Fuite non détectée	Mode de défaillance
Encrassement	Cause

Cas 2	
Attribut	Rôle
Détecteur catharométrique	Capteur utilisé
Méthane	Gaz recherché
Vapeur d'eau	Gaz présent
Vapeur de solvant	Gaz présent
Déclenchement intempestif	Mode de défaillance
Un gaz cause des interférences avec le capteur	Cause

Cas 3	
Attribut	Rôle
Détecteur électrochimique	Capteur utilisé
Hydrogène	Gaz recherché
Faible hygrométrie	Condition ambiante
Fuite non détectée	Mode de défaillance
Dessèchement de la cellule	Cause

Cas 4	
Attribut	Rôle
Détecteur catharométrique	Capteur utilisé
Hydrogène	Gaz recherché
Forte humidité	Condition ambiante
Fuite non détectée	Mode de défaillance
Technologie non adaptée	Cause

TAB. 1 – Cas simples sur la défaillance de capteurs de gaz.

En n'appliquant que la formule 4, le tableau 2 montre les attributs qui ont été trouvés correspondants à chaque attribut de la requête, ainsi que la similarité calculée pour chaque cas source. Nous constatons dans le premier cas la "présence de poussière" qui a été inférée, grâce aux règles d'inférence intégrées, suite à la présence d'un dépoussiéreur humide dans l'environnement.

Requête		Cas 1 (1)		Cas 2 (0,871)		Cas 3 (0,884)		Cas 4 (0,884)	
Attribut	Rôle	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité
Détecteur PID	Capteur utilisé	Détecteur PID	1	Détecteur catharométrique	0,778	Détecteur électrochimique	0,778	Détecteur catharométrique	0,778
Vapeur de solvant	Gaz recherché	Vapeur de solvant	1	Vapeur de solvant (Gaz présent)	1	Hydrogène	0,833	Hydrogène	0,833
Présence de poussière	Condition ambiante	Présence de poussière	1	Vapeur d'eau	0,364	Faible hygrométrie	0,615	Forte humidité	0,615

TAB. 2 – Résultats obtenus en n'appliquant que la formule 4.

Parmi les résultats obtenus, on peut souligner les mauvaises correspondances suivantes :

1. L'attribut "Présence de poussière" ne correspond pas à ce qui a été trouvé dans les derniers trois cas ;
2. Pour la "Vapeur de solvant" (Gaz recherché) de la requête, la "Vapeur de solvant" du deuxième cas a été trouvée alors que c'était un gaz présent sur le site et non pas le gaz recherché (le méthane dans ce cas).

En rajoutant la condition présentée dans la formule 5, nous éliminons la correspondance avec l'attribut "Vapeur d'eau" trouvée pour le deuxième cas, comme le montre le tableau 3 :

Requête		Cas 1 (1)		Cas 2 (0,717)		Cas 3 (0,884)		Cas 4 (0,884)	
Attribut	Rôle	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité
Détecteur PID	Capteur utilisé	Détecteur PID	1	Détecteur catharométrique	0,778	Détecteur électrochimique	0,778	Détecteur catharométrique	0,778
Vapeur de solvant	Gaz recherché	Vapeur de solvant	1	Vapeur de solvant (Gaz présent)	1	Hydrogène	0,833	Hydrogène	0,833
Présence de poussière	Condition ambiante	Présence de poussière	1	---	---	Faible hygrométrie	0,615	Forte humidité	0,615

TAB. 3 – Résultats obtenus en rajoutant la condition de la formule 5.

L'attribut "Présence de poussière" ne doit pas être comparé aux attributs trouvés comme correspondants dans les deux derniers cas (Faible hygrométrie, Forte humidité). Pour éviter ces comparaisons, nous définissons des régions de similarité, ce qui donne les résultats suivants :

Requête		Cas 1 (1)		Cas 2 (0,717)		Cas 3 (0,692)		Cas 4 (0,692)	
Attribut	Rôle	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité
Détecteur PID	Capteur utilisé	Détecteur PID	1	Détecteur catharométrique	0,778	Détecteur électrochimique	0,778	Détecteur catharométrique	0,778
Vapeur de solvant	Gaz recherché	Vapeur de solvant	1	Vapeur de solvant (Gaz présent)	1	Hydrogène	0,833	Hydrogène	0,833
Présence de poussière	Condition ambiante	Présence de poussière	1	---	---	---	---	---	---

TAB. 4 – Résultats obtenus en définissant des régions de similarité.

Bien que les résultats soient améliorés, le deuxième problème souligné tout à l'heure persiste toujours ; *i.e.* on cherche la "Vapeur de solvant" en tant que gaz à détecter et non pas un gaz présent sur le site, et son attribut correspondant dans le deuxième cas devrait être le méthane. Pour cela, les similarités sont recalculées en tenant compte des rôles des attributs, ce qui donne les résultats souhaités dans le tableau 5.

Requête		Cas 1 (1)		Cas 2 (0,692)		Cas 3 (0,692)		Cas 4 (0,692)	
Attribut	Rôle	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité	Attribut	Similarité
Détecteur PID	Capteur utilisé	Détecteur PID	1	Détecteur catharométrique	0,778	Détecteur électrochimique	0,778	Détecteur catharométrique	0,778
Vapeur de solvant	Gaz recherché	Vapeur de solvant	1	Méthane (Gaz recherché)	0,833	Hydrogène	0,833	Hydrogène	0,833
Présence de poussière	Condition ambiante	Présence de poussière	1	---	---	---	---	---	---

TAB. 5 – Résultats obtenus en tenant compte des rôles des attributs.

8 Conclusion et perspectives

Nous avons présenté dans cet article l'architecture de notre plate-forme COBRA permettant d'établir et de traiter des cas hétérogènes. Nous avons également introduit notre approche pour pallier les problèmes de l'hétérogénéité. Cette approche consiste à utiliser les calculs de similarité et à définir des régions de similarité ainsi que des rôles pour les attributs conduisant à des situations ambiguës. Dans cet article, nous avons présenté les processus d'authoring et de remémoration des cas. En ce qui concerne les autres phases de RàPC, nous avons développé le processus de diagnostic, et nous allons faire une évaluation auprès d'experts de l'INERIS. Cette évaluation est envisagée à deux niveaux :

- Le premier niveau concerne l'utilisabilité de la plate-forme, par exemple : à quel point la structure des cas et les processus de raisonnement sont-ils proches de l'activité réelle de l'expert ? Quels sont les concepts à rajouter à l'ontologie de domaine pour pouvoir décrire les nouveaux cas ? L'expert trouve-t-il les propositions d'aide intéressantes ? *etc.*
- Le deuxième niveau concerne les résultats fournis par le système ; *i.e.* la qualité du diagnostic proposé par le système par rapport à certains cas cibles.

Remerciements

Ce travail est financé par le Ministère de l'Écologie, de l'Énergie, du Développement durable et de l'Aménagement du territoire en France. Nous tenons à remercier M. Sébastien Bouchet (INERIS) pour avoir participé au développement de l'ontologie de capteurs de gaz et de la base de cas.

Références

[1] A. Aamodt. Explanation-Driven Case-Based Reasoning. *Lecture Notes In Computer Science*, pages 274–274, 1994.

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

[3] A. Abou Assali, D. Lenne, et B. Debray. Ontology Development for Industrial Risk Analysis. In *IEEE International Conference on Information & Communication Technologies : from Theory to Applications (ICTTA'08)*, pages 1–5, Damascus, Syria, April 2008.

[4] A. Abou Assali, D. Lenne, B. Debray, et S. Bouchet. COBRA : Une plate-forme de RàPC basée sur des ontologies. In *Actes des 20es Journées Francophones d'Ingénierie des Connaissances (IC 2009)*, pages 277–288, Hammamet, Tunisie, may 2009.

[5] R. Bergmann et A. Stahl. Similarity measures for object-oriented case representations. In *Proceedings of the European Workshop on Case-Based Reasoning, EWCBR'98*, 1998.

[6] M. D'Aquin, J. Lieber, et A. Napoli. *Artificial Intelligence : Methodology, Systems, and Applications*, volume Volume 4183/2006 of *Lecture Notes in Computer Science*, chapter Case-Based Reasoning Within Semantic Web Technologies. Springer Berlin / Heidelberg, 2006.

[7] B. Dìaz-Agudo et P.A. Gonzàlez-Calero. An architecture for knowledge intensive CBR systems. *Advances in Case-Based Reasoning–(EWCBR'00). Springer-Verlag, Berlin Heidelberg New York*, 2000.

[8] B. Dìaz-Agudo et P.A. Gonzàlez-Calero. CBROnto : a task/method ontology for CBR. *Procs. of the 15th International FLAIRS*, 2 :101–106, 2002.

[9] B. Dìaz-Agudo, P.A. Gonzàlez-Calero, J.A. Recio-Garcìa, et A.A. Sànchez-Ruiz-Granados. Building CBR systems with jcolibri. *Science of Computer Programming*, 69(1-3) :68–75, 2007. Special issue on Experimental Software and Toolkits.

[10] B. Fuchs et A. Mille. Une modélisation au niveau connaissance du raisonnement à partir de cas. In L'Harmattan, editor, *Ingénierie des connaissances*, 2005.

[11] L. Lamontagne et G. Lapalme. Raisonnement à base de cas textuels : Etat de l'art et perspectives. *Revue d'intelligence artificielle*, 16(3) :339–366, 2002.

[12] D. Oberle. *Semantic Management of Middleware*, volume 1 of *Semantic Web and Beyond*, chapter Ontologies, pages 33–53. Springer US, 2006.

[13] J.A. Recio-Garcìa, B. Dìaz-Agudo, PA Gonzàlez-Calero, et A. Sanchez. Ontology based CBR with jCOLIBRI. *Applications and Innovations in Intelligent Systems*, 14 :149–162, 2006.

[14] J. Renaud, B.C. Morello, B. Fuchs, et J. Lieber. Raisonnement à Partir de Cas 1 : conception et configuration de produits. *Hermes-Lavoisier, February*, 1, 2007.

[15] M.M. Richter. *Case-Based Reasoning on Images and Signals*, volume 73/2008 of *Studies in Computational Intelligence*, chapter Similarity, pages 25–90. Springer Berlin / Heidelberg, 2008.

[16] C.K. Riesbeck et R.C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, 1989.

[17] C. Yang, B. Farley, et B. Orchard. Automated Case Creation and Management for Diagnostic CBR Systems. *Applied Intelligence : The International Journal of Artificial Intelligence, Neural Networks and Complex Problem-Solving Technologies*, 28(1) :17–28, February 2008.

[18] K. Zhang, J. Tang, M. Hong, J. Li, et W. Wei. Weighted Ontology-Based Search Exploiting Semantic Similarity. *Lecture Notes In Computer Science*, 3841 :498, 2006.

Combinaison de Cas fondée sur un opérateur de Fusion de Connaissance

Julien Cojan, Jean Lieber
Orpailleur, LORIA, CNRS, INRIA, Nancy Université,
BP 239, 54506 Vandœuvre-lès-Nancy

Résumé

La fusion contrainte de connaissance vise à produire une base de connaissance qui synthétise un ensemble de bases de connaissance pouvant être mutuellement inconsistantes et qui satisfasse des contraintes d'intégrité imposées. Cette théorie est appliquée ici à la combinaison de cas pour le raisonnement à partir de cas. Cette approche s'avère étendre l'inférence crédible à partir de cas (*credible case-based inference*) de Eyke Hüllermeier. Sous certaines conditions elle est réductible à de la programmation linéaire.

Mots clés : Combinaison de Cas, Fusion Contrainte de Connaissances, Inférence Crédible à partir de Cas.

Cet article est accepté pour publication au congrès ICCBR-09 sous le titre «Belief Merging-based Case Combination».

1 Introduction

Case-based reasoning (CBR [10]) aims at solving problems thanks to a set of previously solved problems accompanied with their solutions (the source cases). At least, two general approaches of CBR exist. The first one consists in the adaptation of a sole retrieved case. The second one consists in the combination of $k \geq 1$ retrieved cases. Actually, case combination is a generalization of case adaptation : this latter is a case combination with $k = 1$.

In [2], an approach to adaptation based on a belief revision operator, the so-called conservative adaptation, is presented. A belief revision operator $\dot{+}$ associates to two knowledge bases A and B a knowledge base $A \dot{+} B$ that entails B and keeps as much as possible from A. Conservative adaptation of a source case $\texttt{Srce}$ by a target case $\texttt{Tgt}$ consists in a revision $\texttt{ctxt}(\texttt{Srce}) \dot{+} \texttt{ctxt}(\texttt{Tgt})$ where $\texttt{ctxt}(\texttt{Srce})$ and $\texttt{ctxt}(\texttt{Tgt})$ are respectively $\texttt{Srce}$ and $\texttt{Tgt}$ interpreted according to the domain knowledge.

Now, belief revision is generalized by integrity constraint (IC) belief merging [8] that integrates several knowledge bases altogether with constraints.

The purpose here is to substitute belief revision by IC belief merging in conservative adaptation to define a case combination approach.

After the introduction of a running example and a preliminary section, the IC belief merging theory is presented in section 4 and its application to case combination in section 5. Section 6 shows that credible case-based inference [5] is a kind of case combination. The computation of belief merging in numerical spaces is studied in section 7, before a related work review and the conclusion.

2 Introduction of the running example

Assume you have an egg allergic guest and you have the experience of some egg-free dishes. Your guest loves chocolate and chocolate mousse would be a perfect desert, even if you don't have an egg-free recipe at

disposal. Several recipes, the source cases $\texttt{Srce}_i$, can be combined to solve the target case $\texttt{Tgt}$:

$\texttt{Tgt}$: egg-free chocolate mousse recipe
$\texttt{Srce}_1$: chocolate mousse recipe
$\texttt{Srce}_2$: egg-free Chantilly recipe
$\texttt{Srce}_3$: egg-free chocolate cream recipe

The expected solution would be to follow the main lines from $\texttt{Srce}_1$ but to substitute the egg-snow by egg-free Chantilly from $\texttt{Srce}_2$ and the chocolate mix –that contains egg yolks– by the chocolate cream from $\texttt{Srce}_3$.

3 Preliminaries

3.1 Set theory notations

A boolean interpretation $\mathcal{I}$ on a set of variables $\mathcal{V} = \{a_1, a_2, \ldots, a_n\}$ is a mapping from $\mathcal{V}$ to the set of boolean values $\mathbb{B} = \{\texttt{true}, \texttt{false}\}$. It can be assimilated to $(x_1, \ldots, x_n) \in \mathbb{B}^n$ where $\mathcal{I}(a_i) = x_i$ for $i = 1$ to n. Thus, the models $[\![f]\!]$ of a formula f –the set of the interpretations satisfying f– is assimilated to a subset of $\mathcal{U} = \mathbb{B}^n$. Following the *principle of irrelevance of syntax*, the formulas are assimilated in this paper to their models and thus to subsets of $\mathcal{U}$. In particular conjunction $\wedge$, entailment $\models$, and logical equivalence $\equiv$ represent intersection $\cap$, inclusion $\subseteq$, and set equality $=$ on subsets of $\mathcal{U}$. More precisely, $[\![f \wedge g]\!] = [\![f]\!] \cap [\![g]\!]$, $f \models g$ iff $[\![f]\!] \subseteq [\![g]\!]$, and $f \equiv g$ iff $[\![f]\!] = [\![g]\!]$.

Propositional logic can then be generalized considering any set $\mathcal{U}$, in particular attribute-values formalisms correspond to sets of the kind $\mathcal{U} = D_1 \times \ldots \times D_n$ where $D_1, \ldots, D_n$ are more elementary sets like integers $\mathbb{Z}$, positive real numbers $\mathbb{R}^+$, etc.

In order to ease the reading, a variable x_i that can take any value from D_i is just written '_', eg. if $n = 3$ and $a_1 \in D_1$, $a_3 \in D_3$, $\{(a_1, \_, a_3,)\} = \{(a_1, x_2, a_3) \,|\, x_2 \in D_2\}$. So, $\mathcal{U} = \{(\_, \_, \_)\}$.

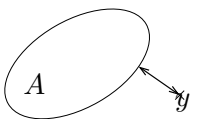
3.2 Metric spaces

Definition 1. *A* pseudo-distance[1] *on a set $\mathcal{U}$ is a function $d : \mathcal{U} \times \mathcal{U} \to [0; +\infty]$ satisfying the separation axiom : for any $x, y \in \mathcal{U}, d(x, y) = 0$ iff $x = y$.*

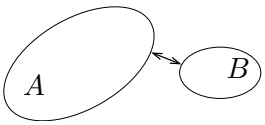
A distance *on $\mathcal{U}$ is a pseudo-distance on $\mathcal{U}$ taking only finite values (for any $x, y \in \mathcal{U}$, $d(x, y) < +\infty$) that satisfies the symmetry axiom : for any $x, y \in \mathcal{U}$, $d(x, y) = d(y, x)$ and the triangular inequality : for any $x, y, z \in \mathcal{U}$, $d(x, z) \leq d(x, y) + d(y, z)$.*

A (pseudo-)metric space *is a pair $(\mathcal{U}, d)$ where d is a (pseudo-)distance on the set $\mathcal{U}$.*

$2^{\mathcal{U}}$ denotes the set of subsets of $\mathcal{U}$. A pseudo-distance d on $\mathcal{U}$ is extended on subsets of $\mathcal{U}$ as follows :



$$d(A, y) = \inf_{x \in A} d(x, y)$$



$$d(A, B) = \inf_{x \in A, y \in B} d(x, y) = \inf_{y \in B} d(A, y)$$

where $A, B \in 2^{\mathcal{U}}$, and $x, y \in \mathcal{U}$ (note that $d : 2^{\mathcal{U}} \times 2^{\mathcal{U}} \to [0; +\infty]$ is not necessary a pseudo-distance).

$A \in 2^{\mathcal{U}}$ is bounded if there exists $K \in \mathbb{R}^+$ such that for each $x, y \in A$, $d(x, y) \leq K$. Given a pseudo-distance on $\mathcal{U}$, $x, y \in \mathcal{U}$ and $r \in [0; +\infty]$, the *right closed ball of center x and radius r* is the set $\mathcal{B}_r^{\mathcal{U}}(x) = \{y \in \mathcal{U} \mid d(x, y) \leq r\}$ and the *left closed ball of center y and radius r* is the set $\mathcal{B}'^{\mathcal{U}}_r(y) = \{x \in \mathcal{U} \mid d(x, y) \leq r\}$. The distinction between these two definitions is relevant when d is not symmetrical, otherwise they are equal.

Definition 2. *A* (pseudo-)discrete metric space *$(\mathcal{U}, d)$ is a (pseudo-)metric space such that, for any $x \in \mathcal{U}$ and $r \in [0; +\infty[$, $\mathcal{B}_r^{\mathcal{U}}(x)$ and $\mathcal{B}'^{\mathcal{U}}_r(x)$ are finite.*

[1] We slightly abuse words here as this may not be in agreement with common definition of pseudo-distance.

This definition is stronger than the usual definition of (pseudo-)discrete metric space which states that for any $x \in \mathcal{U}$, there is an $r > 0$ such that $\mathcal{B}_r^{\mathcal{U}}(x) = \mathcal{B}'^{\mathcal{U}}_r(x) = \{x\}$.

If $(\mathcal{U}, d)$ is discrete then for A, B two subsets of $\mathcal{U}$, if A is bounded, then A is finite and $d(A, B) = d(A, y)$ for some $y \in B$. Moreover $\{z \in \mathcal{U} \mid d(A, z) = 0\} = A$ (i.e. A is closed).

To avoid continuity issues, like no minimum for a distance, only discrete spaces will be considered in the following. In particular $\mathbb{R}$ will be approximated by decimals of a fixed maximum length.

3.3 CBR : Definitions and Hypotheses

Cases and domain knowledge. Case-based reasoning (CBR) aims at solving problems of a given application domain with the help of previous solving episodes, or *cases*. In this paper, these notions are formalized as follows. Let $\mathcal{U}_{\texttt{pb}}$ and $\mathcal{U}_{\texttt{sol}}$ be two sets : $x \in \mathcal{U}_{\texttt{pb}}$ is a *problem instance*, $X \in \mathcal{U}_{\texttt{sol}}$ is a *solution instance*. A *problem* pb is a class of problem instances : $\texttt{pb} \in 2^{\mathcal{U}_{\texttt{pb}}}$. A *solution* sol is a class of solution instances : $\texttt{sol} \in 2^{\mathcal{U}_{\texttt{sol}}}$.

Let $\mathcal{U} = \mathcal{U}_{\texttt{pb}} \times \mathcal{U}_{\texttt{sol}}$. A *case* is a subset of $\mathcal{U}$. A *singleton case* is a case with only one element : $\texttt{Case} = \{(x, X)\}$. Given x and Case, a problem instance and a case, $\Phi_x(\texttt{Case})$ denotes the projection of x on solution instances :

$$\Phi_x(\texttt{Case}) = \{X \in \mathcal{U}_{\texttt{sol}} \mid (x, X) \in \texttt{Case}\}$$

In particular, if $\texttt{Case} = \{(x, X)\}$ is a singleton case then $\Phi_x(\texttt{Case}) = \{X\}$.

The relationship stating that a solution sol solves a problem pb is formalized by a binary relation $\rightsquigarrow$ on $\mathcal{U} = \mathcal{U}_{\texttt{pb}} \times \mathcal{U}_{\texttt{sol}}$: pb is solved by sol if, for every $x \in \texttt{pb}$ exists $X \in \texttt{sol}$ such that $x \rightsquigarrow X$.

$\rightsquigarrow$ is not assumed to be completely known. By contrast, it is assumed that a finite set of cases, the *case base* CB and some domain knowledge are available, and that any case of the case base CB –called a *source case* and denoted by Srce– has the following property : for each $x \in \mathcal{U}_{\texttt{pb}}$, if $\Phi_x(\texttt{Srce}) \neq \emptyset$ then there exists $X \in \Phi_x(\texttt{Srce})$ such that $x \rightsquigarrow X$ (see figure 1).

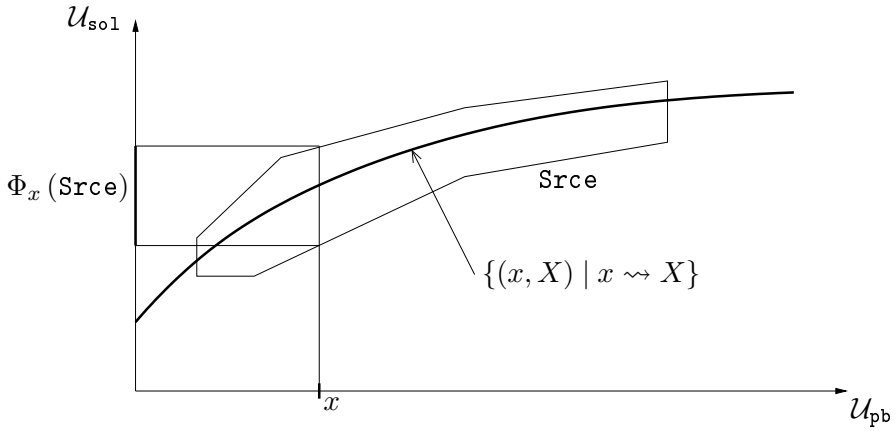


FIG. 1 – A source case Srce.

The target case, denoted by Tgt, is the case for which the solution part has to be made more precise by the current CBR session. In general, before the CBR inference, nothing is known about this solution : $\texttt{Tgt} = \texttt{tgt} \times \mathcal{U}_{\texttt{sol}}$ with $\texttt{tgt} \in 2^{\mathcal{U}_{\texttt{pb}}}$, the *target problem* (Fig. 2). A *singleton target problem* is a target problem with only one element : $\texttt{tgt} = \{x_0\}$.

The *domain knowledge* states that some pairs (x, X) are not licit : $x \not\rightsquigarrow X$. Thus, it corresponds to a necessary condition for $x \rightsquigarrow X$. It is formalized by a subset DK of $\mathcal{U} = \mathcal{U}_{\texttt{pb}} \times \mathcal{U}_{\texttt{sol}}$ and satisfies the implication : $x \rightsquigarrow X$ implies $(x, X) \in \texttt{DK}$ (or, by contraposition, $(x, X) \notin \texttt{DK}$ implies $x \not\rightsquigarrow X$). For Case, a given case, its elements (x, X) that are not consistent with DK have not to be considered (they are known to be illicit). Thus, Case is to be considered in conjunction with DK, i.e. in its context $\texttt{ctxt}(\texttt{Case}) = \texttt{DK} \cap \texttt{Case}$. If no domain knowledge is available, then $\texttt{DK} = \mathcal{U}$: every pair $(x, X) \in \mathcal{U}$ is *a priori* licit.

Case-based inference. Given a target case Tgt, a case base CB and the domain knowledge $\mathtt{DK} \subseteq \mathcal{U}$, the case-based inference aims at proposing a case SolvedTgt that makes Tgt more precise (Fig. 2) :

$$\mathtt{DK} \cap \mathtt{SolvedTgt} \subseteq \mathtt{DK} \cap \mathtt{Tgt} \tag{1}$$

Two main approaches for this inference are described in the CBR literature. The first one is based on adaptation and the second one on combination.

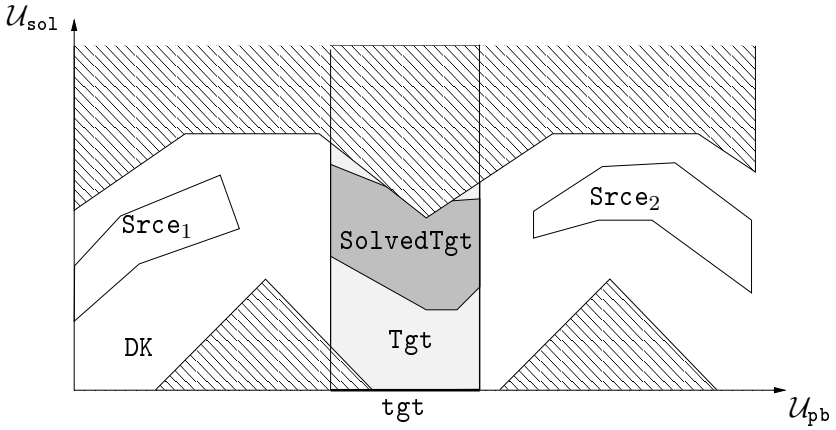


FIG. 2 – Result of a CBR session, Tgt has been specialized into SolvedTgt.

3.4 Formalization of the example

The cooking problem specification consists in the three following conditions : whether the dish is frothy, whether it contains eggs, and whether it contains chocolate. $\mathcal{U}_{\mathtt{pb}} = \mathcal{U}_1 \times \mathcal{U}_2 \times \mathcal{U}_3$.

Only the ingredient amounts and the volume of froth will be considered for the solution. All the values are taken for a single serving –which explains the real values for the eggs number. $\mathcal{U}_{\mathtt{sol}} = \mathcal{U}_4 \times \mathcal{U}_5 \times \ldots \times \mathcal{U}_9$. And finally $\mathcal{U} = \mathcal{U}_{\mathtt{pb}} \times \mathcal{U}_{\mathtt{sol}}$. The cases values are given in table 1.

	attribute	Tgt	Srce$_1$	Srce$_2$	Srce$_3$
$\mathcal{U}_1 = \mathbb{B}$	is frothy	true	true	true	false
$\mathcal{U}_2 = \mathbb{B}$	has eggs	false	true	false	false
$\mathcal{U}_3 = \mathbb{B}$	has chocolate	true	true	false	true
$\mathcal{U}_4 = \mathbb{R}$	froth volume (ml)	_	200	225	0
$\mathcal{U}_5 = \mathbb{R}$	number of eggs	_	0.67	0	0
$\mathcal{U}_6 = \mathbb{R}$	chocolate mass (g)	_	35	0	25
$\mathcal{U}_7 = \mathbb{R}$	cream mass (g)	_	10	0	125
$\mathcal{U}_8 = \mathbb{R}$	sugar mass (g)	_	65	50	_
$\mathcal{U}_9 = \mathbb{R}$	soya volume (ml)	_	0	170	0

TAB. 1 – Formalization of the egg free chocolate mousse example.

The domain knowledge is given by DK :

$$\mathtt{DK} = R_{\mathtt{vFroth}} \cap R_{\mathtt{hasFroth}} \cap R_{\mathtt{hasEggs}} \cap R_{\mathtt{hasChocolate}}$$

where $R_{\mathtt{vFroth}}$ states that the froth obtained from an egg is at most 200 ml and 220 ml from 170 ml of soya milk ($220/170 \simeq 1.32$) :

$$R_{\mathtt{vFroth}} = \{(\_,\_,\_,\mathtt{vFroth},\mathtt{eggs},\_,\_,\_,\mathtt{soya}) \,|\mathtt{vFroth} \leq 200 \times \mathtt{eggs} + 1.32 \times \mathtt{soya}\}$$

$R_{\texttt{hasFroth}}$, $R_{\texttt{hasEggs}}$, and $R_{\texttt{hasChocolate}}$ force `hasFroth` (resp. `hasEggs` and `hasChocolate`) to be true only when there is froth (resp. eggs and chocolate) in the recipe :

$$R_{\texttt{hasFroth}} = \{(\texttt{hasFroth}, \_, \_, \texttt{vFroth}, \_, \_, \_, \_, \_) \mid \texttt{hasFroth} = \texttt{true} \text{ iff } \texttt{vFroth} \neq 0\}$$
$$R_{\texttt{hasEggs}} = \{(\_, \texttt{hasEggs}, \_, \_, \texttt{eggs}, \_, \_, \_, \_) \mid \texttt{hasEggs} = \texttt{false} \text{ iff } \texttt{eggs} = 0\}$$
$$R_{\texttt{hasChocolate}} = \{(\_, \_, \texttt{hasChocolate}, \_, \_, \texttt{chocolate}, \_, \_, \_) \mid \texttt{hasChocolate} = \texttt{false} \text{ iff } \texttt{chocolate} = 0\}$$

Therefore, the fifth component of $\texttt{DK} \cap \texttt{Tgt}$ is $(\texttt{DK} \cap \texttt{Tgt})_5 = 0$.

The following distance d is defined on $\mathcal{U}$, for $x, y \in \mathcal{U}$, by :

$$d(x,y) = d_{\texttt{pb}}(x,y) + d_{\texttt{sol}}(x,y)$$
$$d_{\texttt{pb}}(x,y) = \sum_{i=1}^{3} w_i \times \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$
$$d_{\texttt{sol}}(x,y) = \sum_{i=4}^{9} w_i |y_i - x_i|$$

The choice of the weights w_i reflects the relative importance of the different dimensions. In particular here, the eggs and soya milk are used to generate froth, thus froth's dimension should get a higher importance than egg's and soya's.

4 Integrity Constraint Belief Merging

The following scenario illustrates the notion of integrity constraint belief merging. Let us consider an agent that has some knowledge about the world that he/she considers to be inviolable : this is his/her integrity constraints (IC). Now, he/she receives from several sources some knowledge about the world. Taking the conjunction of all these sources does not necessarily lead to a knowledge base consistent with the IC of the agent.

Various operators may be used to merge these sources of knowledge in a result consistent with the integrity constraints. Such an IC merging operator should satisfy some postulates [8], as it is explained in section 4.1. A straightforward generalization of this work to a more general formalism is presented in section 4.2. Then, an example of IC merging operator is presented (section 4.3).

4.1 IC Merging in Propositional Logic

The definition below is a reformulation from [8], with substitution of propositional formulas on $\mathcal{V}$ by subsets of $\mathbb{B}^n$:

Definition 3. *Let $\mathcal{U} = \mathbb{B}^n$. An* IC merging operator *on $\mathcal{U}$ is a mapping $\triangle : (IC, M) \mapsto \triangle_{IC}(M)$, where $IC \in 2^{\mathcal{U}}$ is the integrity constraint and M –the set of beliefs to be merged– is a finite multi-set of non empty subsets of $\mathcal{U}$, satisfying the following postulates :*

($\triangle$-1) $\triangle_{IC}(M) \subseteq IC$
($\triangle$-2) *If* $IC \neq \emptyset$ *then* $\triangle_{IC}(M) \neq \emptyset$
($\triangle$-3) *If* $\bigcap M \cap IC \neq \emptyset$ *then* $\triangle_{IC}(M) = \bigcap M \cap IC$
($\triangle$-4) *If* $A_1 \subseteq IC$ *and* $A_2 \subseteq IC$ *then* $\triangle_{IC}(\{A_1, A_2\}) \cap A_1 \neq \emptyset$ *iff* $\triangle_{IC}(\{A_1, A_2\}) \cap A_2 \neq \emptyset$
($\triangle$-5) $\triangle_{IC}(M_1) \cap \triangle_{IC}(M_2) \subseteq \triangle_{IC}(M_1 \cup M_2)$
($\triangle$-6) *If* $\triangle_{IC}(M_1) \cap \triangle_{IC}(M_2) \neq \emptyset$ *then* $\triangle_{IC}(M_1 \cup M_2) \subseteq \triangle_{IC}(M_1) \cap \triangle_{IC}(M_2)$
($\triangle$-7) $\triangle_{IC_1}(M) \cap IC_2 \subseteq \triangle_{IC_1 \cap IC_2}(M)$
($\triangle$-8) *If* $\triangle_{IC_1}(M) \cap IC_2 \neq \emptyset$ *then* $\triangle_{IC_1 \cap IC_2}(M) \subseteq \triangle_{IC_1}(M)$

where $IC, IC_1, IC_2, A_1, A_2 \in 2^{\mathcal{U}}$, $A_1 \neq \emptyset$, $A_2 \neq \emptyset$ and M, M_1, and M_2 are three multisets of non empty subsets of $\mathcal{U}$. $M_1 \cup M_2$ denotes the multi-set union of M_1 and M_2.

Note that there is another postulate in [8] that expresses the principle of irrelevance of syntax. By working on interpretations the independence to the syntax is already implied, thus this postulate is reformulated in the following tautology : if $M_1 = M_2$ and $IC_1 = IC_2$ then $\triangle_{IC_1}(M_1) = \triangle_{IC_2}(M_2)$.

IC merging and belief revision. Belief revision is usually presented as the change of an agent belief A after some facts B are known by him. Some beliefs in A may have to be left as being in contradiction with B but others may not have interference and should be kept. The resulting belief $A \dot{+} B$ should entail B and keep "as much as possible" of A. The notion of IC merging can be considered as a generalization of the notion of revision in the sense that if $\dot{+}$ is defined by

$$A \dot{+} B = \triangle_B(\{A\}) \tag{2}$$

with $\triangle$ an IC merging operator, then $\dot{+}$ satisfies the postulates of revision (i.e., the postulates of the "AGM theory" [1], that have been applied to propositional logic in [7]).

4.2 Generalization

The definition of an IC merging operator on a given set $\mathcal{U}$ is the same as definition 3, except that $\mathcal{U}$ is any set, not necessarily $\mathbb{B}^n$.

Definition 4. *A* pre-IC merging operator *on a set $\mathcal{U}$ is a mapping $\triangle : (IC, M) \mapsto \triangle_{IC}(M)$, where $IC \in 2^{\mathcal{U}}$ and M is a finite multi-set of subsets of $\mathcal{U}$, satisfying the postulates $(\triangle\text{-}1)$, $(\triangle\text{-}3)$ to $(\triangle\text{-}8)$, and $(\triangle\text{-}2')$ a weakened version of $(\triangle\text{-}2)$:*

$(\triangle\text{-}2')$ *If $IC \neq \emptyset$ and every set in M is bounded, then $\triangle_{IC}(M) \neq \emptyset$*

Definition 5. *An* IC merging operator *is a pre-IC merging operator that satisfies postulate $(\triangle\text{-}2)$.*

4.3 Example of pre-IC Merging Operator

The operator presented in this section is inspired from one of the DA^2 operators [8].

Let $(\mathcal{U}, d)$ be a pseudo-metric space. Let d^{Σ} be the function associating to the pair (M, y) –where M is a finite multiset of subsets of $\mathcal{U}$, and $y \in \mathcal{U}$– the real number

$$d^{\Sigma}(M, y) = \sum_{A \in M} d(A, y)$$

For $IC \in 2^{\mathcal{U}}$, let $d^{\Sigma}(M, IC) = \inf_{y \in IC} d^{\Sigma}(M, y)$. Let $\triangle^{d,\Sigma}$ be the operator defined as follows :

$$\triangle^{d,\Sigma}_{IC}(M) = \left\{ y \in IC \mid d^{\Sigma}(M, y) = d^{\Sigma}(M, IC) \right\}$$

Proposition 1. *If $(\mathcal{U}, d)$ is discrete, $\triangle^{d,\Sigma}$ satisfies the postulates $(\triangle\text{-}1)$, $(\triangle\text{-}2')$, $(\triangle\text{-}3)$, $(\triangle\text{-}5)$, $(\triangle\text{-}6)$, $(\triangle\text{-}7)$, and $(\triangle\text{-}8)$.*
If d is symmetrical (i.e., it satisfies the symmetry axiom) then $\triangle^{d,\Sigma}$ satisfies $(\triangle\text{-}4)$. It is then a pre-IC merging operator.

A proof for this proposition is given in appendix.

Note, that $(\triangle\text{-}2)$ is not satisfied in general. Consider $\mathcal{U} = \{\log(n) \mid n \in \mathbb{N} \setminus \{0\}\}$ with $d(x, y) = |y - x|$, $IC = \{\log(2p) \mid p \in \mathbb{N} \setminus \{0\}\}$, $M = \{A\}$ with $A = \{\log(2p + 1) \mid p \in \mathbb{N}\}$. Then $\triangle^{d,\Sigma}_{IC}(M) = \emptyset$.

5 Case Combination based on a pre-IC Merging Operator

5.1 Conservative Adaptation

Conservative adaptation [2] is an approach to adaptation based on belief revision. Its principle is to reuse "as much as possible" of $\texttt{Srce}$ while being consistent with $\texttt{Tgt}$. Both $\texttt{Srce}$ and $\texttt{Tgt}$ must be considered according to domain knowledge $\texttt{DK}$. As for belief revision, the meaning of "as much as possible" is variable. The idea is to define conservative adaptation parameterized by a belief revision operator $\dot{+}$:

$$\texttt{SolvedTgt} = (\texttt{DK} \cap \texttt{Srce}) \dot{+} (\texttt{DK} \cap \texttt{Tgt})$$

This inference is called $\dot{+}$-conservative adaptation.

5.2 $\triangle$-combination of cases

Definition. Let $\texttt{SCS} = \{\texttt{Srce}_1, \ldots, \texttt{Srce}_k\}$ be a subset of the case base $\texttt{CB}$. Let $\triangle$ be a pre-IC merging operator on $\mathcal{U} = \mathcal{U}_{\texttt{pb}} \times \mathcal{U}_{\texttt{sol}}$. $\triangle$-combination of cases is a generalization of $\dot{+}$-conservative adaptation : $\dot{+}$ is generalized in $\triangle$ and the sole selected case is generalized in the set of source cases $\texttt{SCS}$. Thus, $\texttt{SolvedTgt} = \texttt{CC}^{\texttt{DK}}_{\triangle}(\texttt{SCS}, \texttt{Tgt})$ with

$$\texttt{CC}^{\texttt{DK}}_{\triangle}(\{\texttt{Srce}_1, \ldots, \texttt{Srce}_k\}, \texttt{Tgt}) = \triangle_{\texttt{DK} \cap \texttt{Tgt}}(\{\texttt{DK} \cap \texttt{Srce}_1, \ldots, \texttt{DK} \cap \texttt{Srce}_k\}) \quad (3)$$

I.e., the contribution of the source cases are merged in a result that specializes the target case.

If $k = \text{card}(\texttt{SCS}) = 1$, then $\texttt{CC}^{\texttt{DK}}_{\triangle}(\texttt{SCS}, \texttt{Tgt})$ is a $\dot{+}$-conservative adaptation, with $\dot{+}$ defined by (2).

Properties. In this section, the consequences of the postulates of (pre-)IC merging operators are discussed from a $\triangle$-combination of cases viewpoint.

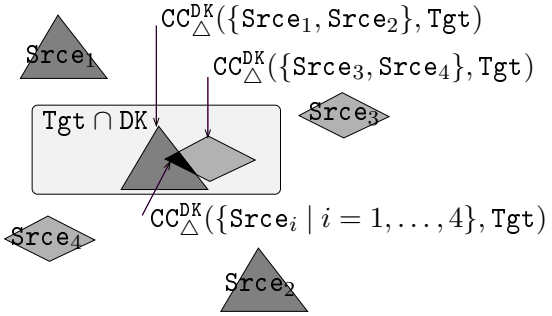
$(\triangle\text{-}1)$ entails that $\texttt{DK} \cap \texttt{SolvedTgt} \subseteq \texttt{DK} \cap \texttt{Tgt}$ which is the property (1) required for the case-based inference (cf. section 3.3).

$(\triangle\text{-}2')$ entails that if the target case is consistent with the domain knowledge –$\texttt{DK} \cap \texttt{Tgt} \neq \emptyset$– and each source case is bounded, –e.g., singleton cases– then $\texttt{DK} \cap \texttt{SolvedTgt}$ is satisfiable. $(\triangle\text{-}2)$ is stronger as it does not require the source cases to be bounded.

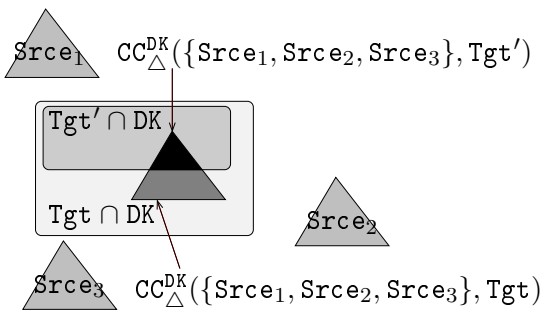
$(\triangle\text{-}3)$ entails that if the cases of $\texttt{SCS}$ are consistent altogether with $\texttt{DK}$, then $\texttt{SolvedTgt}$ is the conjunction of $\texttt{DK}$, $\texttt{Tgt}$, and every $\texttt{Srce} \in \texttt{SCS}$.

$(\triangle\text{-}4)$ enforces equity between the source cases : if two source cases are consistent with the target context, then either both of them are taken into account in the combination or none of them.

$(\triangle\text{-}5)$ to $(\triangle\text{-}8)$ characterize the maximal preservation of the source cases according to the local decomposition of $\texttt{SCS}$ and $\texttt{Tgt}$.



$(\triangle\text{-}5)$ and $(\triangle\text{-}6)$ state that if the combination of two subsets of $\texttt{SCS}$ provide consistent solutions, then the combination of the whole is the conjunction of both solutions.



$(\triangle\text{-}7)$ and $(\triangle\text{-}8)$ state that if $\texttt{Tgt}$ is specialized into $\texttt{Tgt}'$ that is consistent with the case combination for $\texttt{Tgt}$, then the case combination for $\texttt{Tgt}'$ is obtained by conjunction with $\texttt{Tgt}'$.

5.3 Application to the example

Consider the merging operator defined in section 4.3 using the distance d defined in section 3.4. The values of dimensions $\mathcal{U}_i$ for $i = 1, 2, 3, 5$ are fixed in $\texttt{DK} \cap \texttt{Tgt}$, so the space to be explored for the minima of d^Σ corresponds to $i = 4, 6, 7, 8, 9$.

From the structure of d, it can be shown that the minimum can be searched independently for dimensions $\mathcal{U}_i$ with $i = 6, 7, 8$ as there is no constraint relating their values. Then the minima does not depend on the weight w_i, it is reached for :

$$x_6 = 25 \qquad x_7 = 10 \qquad 50 \leq x_8 \leq 65$$

$\mathcal{U}_4$, $\mathcal{U}_5$, and $\mathcal{U}_9$ are related through $R_{\texttt{vFroth}}$, the search must then follow this restriction : $x_4 \leq 200 \times x_5 + 1.32 \times x_9$. As $x_5 = 0$ it becomes : $x_4 \leq 1.32 \times x_9$.

As seen in section 3.4, the volume of froth should be given priority over the ways to making it. From the structure of d^Σ used in $\triangle^{d,\Sigma}$, it can be shown that the condition $w_4 > 3 \times (w_9 + w_5)$ is enough to ensure that priority (3 for the number of cases and w_9, w_5 because these are the dimensions in competition). Under this assumption, the minima is obtained for :

$$x_4 = 200 \qquad x_9 = 165$$

This result matches with what was expected in section 2 : the froth volume and chocolate mass are close to those of $\texttt{Srce}_1$. The use of soya milk to generate froth is taken from $\texttt{Srce}_2$ with an adaptation according to the froth volume. $\texttt{Srce}_3$ had little influence, however its selection as source case offsets the absence of chocolate in $\texttt{Srce}_2$.

6 Application to CCBI

6.1 Credible Case-Based Inference

Assumptions. Credible case-based inference (CCBI [5]) is an approach to CBR for which the problem-solution relation is assumed to be a partial function : if $x \rightsquigarrow X$ and $x \rightsquigarrow X'$ then $X = X'$. Moreover, each source case is a singleton $\texttt{Srce}_i = \{(x_i, X_i)\}$ and the target case is specified by a singleton target problem : $\texttt{Tgt} = \{x_0\} \times \mathcal{U}_{\texttt{sol}}$.

CCBI is based on the idea that the CBR principle "Similar problems have similar solutions" can be modeled thanks to $d_{\texttt{pb}}$, a symmetrical pseudo-distance on $\mathcal{U}_{\texttt{pb}}$, $d_{\texttt{sol}}$, a symmetrical pseudo-distance on $\mathcal{U}_{\texttt{sol}}$, and h, a *similarity profile*, i.e., a function $h : [0; +\infty] \to [0; +\infty]$ such that for *most* $x, y \in \mathcal{U}_{\texttt{pb}}$, if $x \rightsquigarrow X$ and $y \rightsquigarrow Y$ then

$$d_{\texttt{sol}}(X, Y) \leq h(d_{\texttt{pb}}(x, y)) \tag{4}$$

Thus, the similarity between solutions is constrained by the similarity between problems.[2] A way to learn h from the case base is also described in [5] and it is proven, under technical assumptions, that the probability of having the constraint (4) violated converges to 0 as the size of the case base grows.

Definition of CCBI. Given a set $\texttt{SCS}$ of source cases $\texttt{Srce}_i = \{(x_i, X_i)\}$ and a target problem x_0, if $x = x_i$ and $y = x_0$ satisfies (4) for any i, then the solution X_0 of x_0 satisfies $d_{\texttt{sol}}(X_i, X_0) \leq h(d_{\texttt{pb}}(x_i, x_0))$. In other words (Fig. 3) :

$$X_0 \in \mathcal{C}_{\text{CCBI}} = \bigcap_i \mathcal{B}^{\mathcal{U}_{\texttt{sol}}}_{h(d_{\texttt{pb}}(x_i, x_0))}(X_i) \tag{5}$$

Therefore, $\texttt{SolvedTgt} = \{x_0\} \times \mathcal{C}_{\text{CCBI}}$ solves $\texttt{Tgt}$.

This inference is only credible and not certain since (4) is only satisfied for most $x, y \in \mathcal{U}_{\texttt{pb}}$.

[2] In fact, CCBI is introduced in [5] thanks to similarity measures $\mathcal{S}_{\texttt{pb}}$ and $\mathcal{S}_{\texttt{sol}}$ on $\mathcal{U}_{\texttt{pb}}$ and $\mathcal{U}_{\texttt{sol}}$, but the definition presented in the current paper is equivalent. Indeed, a similarity measure $\mathcal{S}$ on $\mathcal{U}$ verifying $\mathcal{S}(x, y) = 1$ iff $x = y$ can be defined thanks to a pseudo-distance d on $\mathcal{U}$ by $\mathcal{S}(x, y) = \frac{1}{1+d(x,y)}$ and vice-versa.

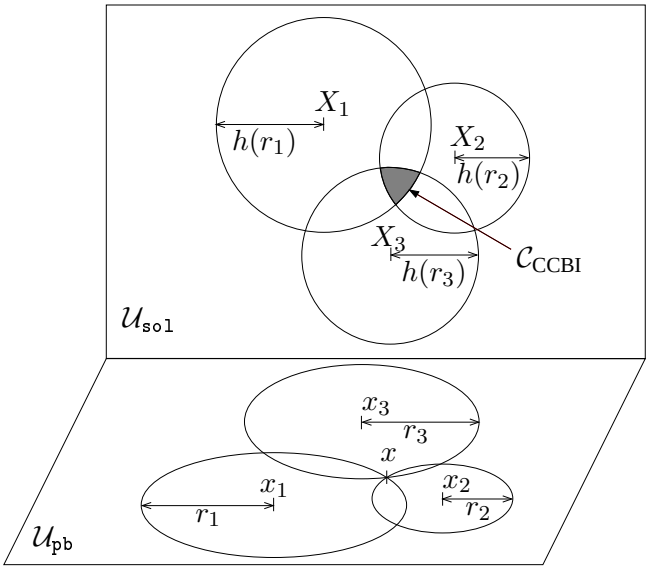


FIG. 3 – Credible Case-Based Inference.

6.2 $\triangle^{d,\Sigma}$-combination of cases extends CCBI

Proposition 2. *CCBI assumption about the case base is made (cf. section 6.1).*
Let d be the pseudo-distance on $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{sol}$ defined for $(x, X), (y, Y) \in \mathcal{U}$ by

$$d((x, X), (y, Y)) = \max\{h(d_{pb}(x, y)), d_{sol}(X, Y)\}$$

Let $\mathcal{C}_{CCBI}$ be the result of CCBI, and $\mathcal{C}_{\triangle^{d,\Sigma}}$ be the result of the $\triangle^{d,\Sigma}$-combination of cases without domain knowledge ($DK = \mathcal{U}$) :

$$\mathcal{C}_{CCBI} = \bigcap_{i=1}^{n} \mathcal{B}^{\mathcal{U}_{sol}}_{h(d_{pb}(x_i,x_0))}(X_i)$$
$$\mathcal{C}_{\triangle^{d,\Sigma}} = \Phi_{x_0}\left(\mathcal{CC}^{\mathcal{U}}_{\triangle^{d,\Sigma}}(\mathcal{SCS}, \mathcal{Tgt})\right)$$

If CCBI provides a consistent result –$\mathcal{C}_{CCBI} \neq \emptyset$– then it coincides with the $\triangle^{d,\Sigma}$-case combination :

$$\mathcal{C}_{CCBI} = \mathcal{C}_{\triangle^{d,\Sigma}}$$

A proof for this proposition is given in appendix.

7 Computing IC merging in numerical spaces

The computation of the merging $\triangle^{d,\Sigma}_{IC}(M)$ is considered in this section with the assumptions :
– $\mathcal{U} = D_1 \times \ldots \times D_n$ with D_i an interval of $\mathbb{Z}$ or of $\mathbb{R}$.
– For $x, y \in \mathcal{U}$ $d(x, y) = \sum_{i=1}^{n} w_i |y_i - x_i|$.
– IC can be defined by a finite set of linear inequalities.
– $M = \{A_1, \ldots, A_p\}$ and every A_j can also be defined by a finite set of linear inequalities.

The computation of $\triangle^{d,\Sigma}_{IC}(M)$ is equivalent to the minimization of the function $y \mapsto d^{\Sigma}(M, y)$ under the constraint $y \in IC$.

Proposition 3. *The computation of $\triangle^{d,\Sigma}_{IC}(M)$ is reducible to a linear programming problem [6].*

Sketch of proof. Minimizing $d^{\Sigma}(M, y)$ is reducible to minimizing $\sum_{j=1}^{p}\sum_{i=1}^{n} w_i|y_i - x_i^j|$ under the constraints $x^j \in A_j$. It is itself reducible to minimizing $\sum_{j=1}^{p}\sum_{i=1}^{n} w_i z_i^j$ with the additional constraints : $z_i^j \geq y_i - x_i^j$ and $z_i^j \geq x_i^j - y_i$, which is a linear programming problem. □

This property shows that if every D_i is an interval of $\mathbb{R}$, i.e. it is reducible to a real linear programming problem, then the computation cost is polynomial in $n \times p$. If any D_i is a subset of $\mathbb{Z}$ it is a mixed integer linear programming (which is an NP-hard problem).

In particular, the running example of this paper has been computed this way (cf. section 5.3) : the boolean dimensions are replaced by the integer interval $[0, 1]_{\mathbb{Z}} = \{0, 1\}$.

8 Conclusion, Related work, and Future work

The main contribution of this paper is to define an approach to case combination based on an IC belief merging operator. It can be applied to case adaptation as a particular case combination. It is shown to extend credible case-based inference. This approach is, *a priori*, applicable to any formalism on which a pre-IC merging operator can be defined, eg. a pseudo-metric space. Provided that cases are represented by numerical attributes and that the constraints and the distance are linear, belief merging can be reduced to linear programming. The complexity is then polynomial if there are only real value attributes (linear programming) and NP-hard otherwise (mixed integer linear programing). In propositional logic, IC merging and thus $\triangle$-combination is NP-hard, see [8].

Ongoing works are the implementation of a case combination based on an IC merging operator as defined in section 7, with the purpose of experimentation. The possibility to reduce other merging operators to linear programming should be investigated too.

As future work a systematic comparison with other case combination approaches should be performed. The convergence of these approaches should be investigated to determine in particular which ones can be covered by an IC merging operator. The following criteria –inspired by the case combination approaches review given in [4]– can guide the comparison : how is structured the participation of each source case, whether the source cases are reused simultaneously or iteratively, and how the consistency is maintained.

Different ways of structuring the combination exist. Static structures as in Decentralized CBR (DzCBR) [3] where a set of contexts (or viewpoints) is set for the system. Every context generates a local solution according to its local domain knowledge and adaptation knowledge. Structure contained in cases as in DÉJÀ VU where the problem solving episodes are decomposed into a hierarchy of cases from the most abstract one that gives the main frame of the solution to the most concrete ones that solve subproblems. Coverage of the target case by a set of source cases as in IDIOM [11] and COMPOSER [9] –a source case represent a partial solution with constraints for its inclusion in a global solution. In the approach presented in this paper all the source cases are equally considered, no explicit structure appears.

While COMPOSER, DzCBR, and the approach presented in this paper reuse the cases simultaneously, DÉJÀ VU and IDIOM do it iteratively. In DÉJÀ VU the resolution of a new query starts from the reuse of an abstract source case and is iterated on the resulting subproblems. In IDIOM a solution is built by iteratively incorporating source cases. A further investigation could be to investigate the possibility to express this approach to an iteration of conservative adaptation[3] and to relate it to a combination based on an IC merging operator.

Finally the approaches can be distinguished by the way the consistency is maintained. In DzCBR bridge rules between the contexts enforce the coherence of the local solutions altogether to form a global solution. IDIOM and COMPOSER use a conflict resolution algorithm. In our approach the inconsistencies between cases are managed by an IC merging operator. This motivates the investigation of relationships between conflict resolution and IC merging.

[3] I.e.. first computing $\texttt{SolvedTgt}_1 = \texttt{ctxt}(\texttt{Srce}_1) \dot{+} \texttt{ctxt}(\texttt{Tgt})$ and then iteratively $\texttt{SolvedTgt}_{i+1} = \texttt{ctxt}(\texttt{Srce}_i) \dot{+} \texttt{ctxt}(\texttt{SolvedTgt}_i)$.

Appendix

Proof of property 1

($\triangle$-1) The satisfaction of ($\triangle$-1) is straightforward from the definition of $\triangle_{IC}^{d,\Sigma}(M)$.

($\triangle$-3) If $\bigcap M \cap IC \neq \emptyset$, let y be an element of $\bigcap M \cap IC$, $y \in A$ for any $A \in M$, thus $d(A,y) = 0$ and $d^\Sigma(M,y) = 0$. So $d^\Sigma(M, IC) = 0$ and $y \in \triangle_{IC}^{d,\Sigma}(M)$.

On the other way round, if $y \in \triangle_{IC}^{d,\Sigma}(M)$, then $d^\Sigma(M,y) = 0$ and $d(A,y) = 0$ for any $A \in M$ ($d(A,y) \geq 0$ for any A). The discretion assumption of $(\mathcal{U}, d)$ implies then that $y \in A$. Indeed, consider the set $\{x \in A \mid d(x,y) \leq 1\} \subseteq \mathcal{B}'^{\mathcal{U}}_1(y)$, it is finite and since $d(A,y) < 1$ it is not empty, thus $d(A,y)$ is reached for an $x \in A$. $d(x,y) = 0$ which implies by the separation assumption that $y \in A$. Finally, $y \in \bigcap M$ and by definition of $\triangle_{IC}^{d,\Sigma}(M)$ $y \in IC$ which entails the result.

($\triangle$-5) and ($\triangle$-6) Their satisfaction is obvious when $\triangle_{IC}^{d,\Sigma}(M_1) \cap \triangle_{IC}^{d,\Sigma}(M_2) = \emptyset$.

In the contrary, let y be an element of $\triangle_{IC}^{d,\Sigma}(M_1) \cap \triangle_{IC}^{d,\Sigma}(M_2)$,

$$\begin{aligned} d^\Sigma(M_1, IC) + d^\Sigma(M_2, IC) &= d^\Sigma(M_1, y) + d^\Sigma(M_2, y) \\ &\geq \inf_{z \in IC} \left(d^\Sigma(M_1, z) + d^\Sigma(M_2, z)\right) \geq d^\Sigma(M_1 \cup M_2, IC) \end{aligned}$$

However,

$$\begin{aligned} d^\Sigma(M_1, IC) + d^\Sigma(M_2, IC) &= \inf_{z \in IC} d^\Sigma(M_1, z) + \inf_{z \in IC} d^\Sigma(M_2, z) \\ &\leq \inf_{z \in IC} \left(d^\Sigma(M_1, z) + d^\Sigma(M_2, z)\right) \leq d^\Sigma(M_1 \cup M_2, IC) \end{aligned}$$

Thus all the inequalities can be replace by equalities, in particular the lower bound of $d^\Sigma(M_1, z) + d^\Sigma(M_2, z)$ for $z \in IC$ is $d^\Sigma(M_1, y) + d^\Sigma(M_2, y) = d^\Sigma(M_1 \cup M_2, IC)$, and as for $i = 1, 2$ $d^\Sigma(M_i, z) \geq d^\Sigma(M_i, y)$ this lower bound is reached when $d^\Sigma(M_i, z) = d^\Sigma(M_i, y) = d^\Sigma(M_i, IC)$, ie. $z \in \triangle_{IC}^{d,\Sigma}(M_1 \cup M_2)$ iff $z \in \triangle_{IC}^{d,\Sigma}(M_1) \cap \triangle_{IC}^{d,\Sigma}(M_2)$.

($\triangle$-7) and ($\triangle$-8) Similarly, if $\triangle_{IC_1}^{d,\Sigma}(M) \cap IC_2 = \emptyset$, the result is obvious.

Otherwise, let y be an element of $\triangle_{IC_1}^{d,\Sigma}(M) \cap IC_2$.

$$\begin{aligned} d^\Sigma(M, IC_1) = \inf_{z \in IC_1} d^\Sigma(M, z) \leq \inf_{z \in IC_1 \cap IC_2} d^\Sigma(M, z) = d^\Sigma(M, IC_1 \cap IC_2) \\ d^\Sigma(M, y) = d^\Sigma(M, IC_1) \leq d^\Sigma(M, IC_1 \cap IC_2) \leq d^\Sigma(M, y) \end{aligned}$$

Thus $d^\Sigma(M, IC_1) = d^\Sigma(M, IC_1 \cap IC_2)$ and $\triangle_{IC_1}^{d,\Sigma}(M_1) \cap IC_2 = \triangle_{IC_1 \cap IC_2}^{d,\Sigma}(M_1)$.

($\triangle$-2′) Assume $IC \neq \emptyset$ and every set in M is bounded.

If $M = \emptyset$ then $\bigcap M = \mathcal{U}$ and according to postulate ($\triangle$-3), $\triangle_{IC}^{d,\Sigma}(M) = IC \neq \emptyset$.

If $M \neq \emptyset$, then there exists an $A \in M$, $A \neq \emptyset$ and is bounded so finite. Let $r = d^\Sigma(M, IC) + 1$ and consider $S = \{y \in IC \mid d^\Sigma(M,y) \leq r\}$. S is finite, indeed $S \subseteq \{y \in \mathcal{U} \mid d^\Sigma(A,y) \leq r\} = \bigcup_{a \in A} \mathcal{B}_r^{\mathcal{U}}(a)$ which is a finite disjunction of finite sets. Moreover $S \neq \emptyset$ as $IC \neq \emptyset$ and the lower bound of $x \mapsto d^\Sigma(x, M)$ on IC and S are equal. S being finite this lower bound is reached and $\triangle_{IC}^{d,\Sigma}(M) \neq \emptyset$.

($\triangle$-4) Assume d is symmetrical and consider three sets IC, $A_1 \subseteq IC$ and $A_2 \subseteq IC$.

If $\triangle_{IC}^{d,\Sigma}(\{A_1, A_2\}) \cap A_1 \neq \emptyset$, let y_1 be an element of this set.

$$d^\Sigma(\{A_1, A_2\}, IC) = d^\Sigma(\{A_1, A_2\}, y_1) = d(A_1, y_1) + d(A_2, y_1)$$

as $y_1 \in A_1$, $d(A_1, y_1) = 0$ thus $d^\Sigma(\{A_1, A_2\}, IC) = d(A_2, y_1)$. From the discretion assumption, as $A_2 \neq \emptyset$ there is $y_2 \in A_2$ such that $d(y_1, y_2) = d(y_2, y_1) = d(A_2, y_1)$.

$$d^\Sigma(\{A_1, A_2\}, IC) = d(y_1, y_2) \geq d(A_1, y_2) = d^\Sigma(\{A_1, A_2\}, y_2)$$

As $y_2 \in IC$ $d^{\Sigma}(\{A_1, A_2\}, y_2) \geq d^{\Sigma}(\{A_1, A_2\}, IC)$ thus, there is equality and $y_2 \in \triangle_{IC}^{d,\Sigma}(\{A_1, A_2\})$ which entails that $\triangle_{IC}^{d,\Sigma}(\{A_1, A_2\}) \cap A_2 \neq \emptyset$.
The other implication is symmetrical. □

Proof of property 2

For $X \in \mathcal{C}_{\text{CCBI}}$, $d_{\mathtt{sol}}(X_i, X) \leq h(d_{\mathtt{pb}}(x_i, x_0))$ for any $1 \leq i \leq n$, so

$$d((x_i, X_i), (x_0, X)) = h\left(d_{\mathtt{pb}}(x_i, x_0)\right) \leq \min_{Y \in \mathcal{U}_{\mathtt{sol}}} d((x_i, X_i), (x_0, Y))$$

and

$$\begin{aligned} d^{\Sigma}(\mathtt{CB}, (x_0, X)) &= \sum_{i=1}^{n} d((x_i, X_i), (x_0, X)) \leq \sum_{i=1}^{n} \left(\min_{Y \in \mathcal{U}_{\mathtt{sol}}} d((x_i, X_i), (x_0, Y)) \right) \\ &\leq \min_{Y \in \mathcal{U}_{\mathtt{sol}}} \left(\sum_{i=1}^{n} d((x_i, X_i), (x_0, Y)) \right) = \min_{(x_0, Y) \in \mathcal{U}} d^{\Sigma}(\mathtt{CB}, (x_0, Y)) \end{aligned}$$

Thus, $(x_0, X) \in \mathtt{CC}^{\mathcal{U}}_{\triangle^{d,\Sigma}}(\mathtt{CB}, \mathtt{Tgt})$ and $X \in \mathcal{C}_{\triangle^{d,\Sigma}}$, which shows that $\mathcal{C}_{\text{CCBI}} \subseteq \mathcal{C}_{\triangle^{d,\Sigma}}$.

Moreover, if $\mathcal{C}_{\text{CCBI}} \neq \emptyset$, (ie. there is such an X), then $\min_{Y \in \mathcal{U}_{\mathtt{sol}}} d^{\Sigma}(\mathtt{CB}, (x_0, Y)) \leq d^{\Sigma}(\mathtt{CB}, (x_0, X))$ and the previous inequalities are equalities. Thus, if $Y \in \mathcal{U}_{\mathtt{sol}}$ minimizes $d^{\Sigma}(\mathtt{CB}, (x_0, Y))$, then $\sum_{i=1}^{n} d((x_i, X_i), (x_0, Y)) = \sum_{i=1}^{n} d((x_i, X_i), (x_0, X))$. As $d((x_i, X_i), (x_0, Y)) \geq d((x_i, X_i), (x_0, X))$ for any $1 \leq i \leq n$,

$$d((x_i, X_i), (x_0, Y)) = d((x_i, X_i), (x_0, X)) = h\left(d_{\mathtt{pb}}(x_i, x_0)\right)$$

ie. $Y \in \mathcal{C}_{\text{CCBI}}$, which shows that $\mathcal{C}_{\triangle^{d,\Sigma}} \subseteq \mathcal{C}_{\text{CCBI}}$. □

Références

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

[2] J. Cojan et J. Lieber. Conservative adaptation in metric spaces. In *Proc. of ECCBR-08* :135–149, 2008.

[3] Mathieu d'Aquin, Jean Lieber, et Amedeo Napoli. Decentralized case-based reasoning for the semantic web. In *International Semantic Web Conference*, pages 142–155, 2005.

[4] F. Gebhardt, A. Voß, W. Gräther, et B. Schmidt-Belz. *Reasoning with complex cases*. Kluwer, Boston, 1997.

[5] E. Hüllermeier. Credible Case-Based Inference Using Similarity Profiles. *IEEE Transaction on Knowledge and Data Engineering*, 19(6) :847–858, 2007.

[6] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4) :373–396, 1984.

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

[8] S. Konieczny, J. Lang, et P. Marquis. DA^2 merging operators. *Artificial Intelligence*, 157(1-2) :49–79, 2004.

[9] L. Purvis et P. Pu. An Approach to Case Combination. In A. Voß, editor, *Proc. of the ECAI'96 Workshop : Adaptation in Case-Based Reasoning*, pages 43–46, 1996.

[10] C. K. Riesbeck et R. C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[11] I. Smith, C. Lottaz, et B. Faltings. Spatial composition using cases : IDIOM. In *ICCBR-95 Proceedings*, pages 88–97.

Éléments de réflexion sur les composants d'ontologies et leur manipulation par RàPC

Béatrice Fuchs[1] and Amedeo Napoli[2]
[1] LIRIS, Universités de Lyon 1, Nautibus, 8 bd Niels Bohr, F-69100 Villeurbanne
(bfuchs@liris.univ-lyon1.fr)
[2] LORIA, Bâtiment B, BP 239 F-54506 Vandœuvre-lès-Nancy
(Amedeo.Napoli@loria.fr)

Résumé

Dans cet article, nous proposons quelques premiers éléments de réflexion sur les patrons de conception et les composants d'ontologies, et sur les liens existant avec le raisonnement à partir de cas. Les patrons d'ontologies permettent de guider la conception d'une ontologie et jouent encore le rôle de guides de bonnes pratiques. La conception d'une ontologie est une tâche complexe et nous analysons différentes façons de détecter ce que peut être un patron d'ontologie, la façon de l'adapter et de la réutiliser. Cet article est une première ébauche de réflexion qui demande à être précisé et étendu, notamment avec des exemples pratiques opérationnels, voire de nouvelles propositions de patrons de conception d'ontologie.

1 Introduction

Le Web sémantique peut se concevoir comme un Web où les "agents logiciels" communiquent avec d'autres agents logiciels pour rendre des services et résoudre des problèmes pour les "agents humains" [16]. Ainsi, le Web sémantique se veut un espace de communication et d'interactions où les agents logiciels tirent parti des ressources disponibles pour traiter les problèmes. Parmi les ressources disponibles, il faut distinguer les documents de toutes sortes et les "ontologies". Les ontologies établissent une terminologie commune entre les agents logiciels et humains leur permettant de partager un même point de vue et un même sens sur les concepts et relations manipulés. C'est pourquoi les ontologies prennent une très grande place dans de nombreux domaines liés plus ou moins directement au Web sémantique, comme la gestion des connaissances, les systèmes collaboratifs et notamment les Wikis sémantiques [9, 21], l'intégration de données, la recherche d'information et le commerce électronique.

Du point de vue de la représentation des connaissances, une ontologie matérialise un "modèle" d'un domaine particulier du monde réel. Pratiquement, un tel modèle se représente comme un ensemble de définitions de concepts munis d'attributs et de relations entre ces concepts. L'utilisation effective d'une ontologie nécessite un langage de représentation des ontologies — de la famille des langages de représentation des connaissances — et des modules associés de raisonnement [3]. S'il est important de disposer d'ontologies, il est néanmoins difficile de les construire [29]. Trois grandes stratégies de conception peuvent être considérées :

- la conception manuelle,
- la réutilisation d'ontologies existantes [15],
- la conception semi-automatique d'ontologies à partir de ressources (par l'intermédiaire de méthodes de fouille de données ou de fouille de textes [5, 4]).

Pour ces trois stratégies, il s'avère utile d'être guidé lors de la conception à l'image de ce qui se pratique en génie logiciel, où la production de code s'appuie l'utilisation de "patrons de conception" et la réutilisation de "composants" [17]. Les principes appliqués au logiciel peuvent être ici repris et adaptés pour donner naissance à des *patrons de conception d'ontologies* ou ODP pour *Ontology Design Patterns* [7, 18, 19]. Comme en ingénierie du logiciel un patron propose une solution générique et réutilisable pour un problème de conception, un ODP propose un modèle — implanté en OWL — pour résoudre un problème local de conception d'ontologie. Par exemple, parmi les ODP se distinguent les "patrons logiques" dont certains recouvrent des opérations

logiques élémentaires comme la conjonction et la disjonction ou d'autres des modes d'inférences comme le modus ponens pour faire émerger des connaissances implicites dans une hiérarchie de concepts, les "patrons d'architecture" pour factoriser et contrôler l'architecture de la hiérarchie des concepts, et des "patrons de contenu" comme la composition (et objets composites, ce qui sera notre exemple courant par la suite), les relations n-aires et les séquences (d'actions).

D'assez nombreux parallèles peuvent être faits entre la recherche et la mise en œuvre de patrons de conception et le raisonnement à partir de cas (abrégé en RÀPC dans la suite) comme évoqué dans [7, 6]. Auparavant, il nous faut faire une différence entre "composant d'ontologie" et "patron de conception d'ontologie" :

- Un composant dénote un bloc élémentaire qui va entrer dans la conception d'une ontologie et qui peut être très spécifique, comme par exemple la description d'une molécule spécifique en chimie organique [25].
- Un patron de conception dénote un bloc générique qui va entrer dans la conception de plusieurs ontologies ou encore qui va servir de base via une adaptation à la création de plusieurs composants, comme par exemple la structure de graphe peut servir de base à la description d'une molécule mais aussi à d'autres structures composites.

Les patrons d'ontologies sont plutôt en petit nombre et donc s'ils se retrouvent assez facilement ils doivent être le plus souvent adaptés. Curieusement un lien direct et assez naturel peut être établi avec le système CADRE [20] où quatre cas génériques d'architecture de bâtiments étaient fournis mais où tout l'effort portait sur le processus d'adaptation de ces cas (en fonction de connaissances du domaine disponibles). Le lien entre architecture et patrons de conception est d'ailleurs explicitement mentionné dans [19].

Le but des éléments de réflexion proposés ici est de faire un point sur les rapports entre conception d'ontologies et techniques de RÀPC, plus particulièrement entre les composants d'ontologies, les patrons de conception d'ontologies et le processus d'adaptation dans la conception d'ontologies. Des liens seront faits avec certaines manières de produire des composants et des patrons, qui sont essentiellement manuelles ou semi-automatiques. Ainsi, un certain nombre de travaux en découvertes de connaissances à partir de documents textuels ou de ressources hétérogènes peuvent être repris et étendus dans cette voie comme par exemple [5, 4].

Cet article est une première ébauche de réflexion qui demande à être précisé et étendu, notamment avec des exemples pratiques opérationnels, voire de nouvelles propositions de patrons de conception d'ontologie. Pour simplifier le texte et lorsqu'il n'y a pas de différence à faire entre patron et composant, nous parlerons simplement de "composant d'ontologie".

2 Quelques problèmes liés à la conception d'ontologies

Dans ce qui suit, nous considérons qu'une ontologie fournit un modèle explicite, non ambigu et opérationnel d'un certain domaine. Une ontologie est codée dans un formalisme de représentation des connaissances comme une logique de descriptions ou OWL [3]. Ce qui nous intéresse en premier lieu ici est ce qui peut être nommé le *schéma d'ontologie*, qui est constitué de l'ensemble des concepts et des relations qui existent entre eux, à la façon d'un réseau sémantique. Les attributs propres d'un concept tout comme le caractère primitif ou défini du concept ne sont pas nécessairement pris en compte dans ce schéma.

Il y a plusieurs façons de considérer les composants et patrons de conception d'ontologies. Tout d'abord, comme pour l'acquisition de connaissances à partir d'expertise, un patron peut se voir comme un bloc de conception générique, comme l'expression d'un savoir-faire ou encore un guide de bonne pratique pour la conception [18, 19]. De tels patrons interviennent dans la mise en forme et la concision de l'ontologie et joue un rôle tangible sur la qualité de l'ontologie résultante. Dans l'industrie manufacturière par exemple, ces patrons prennent souvent la forme de "gabarits". En parallèle et plus en relation avec la découverte de connaissances, un composant — qui n'est pas nécessairement un patron de conception — peut se voir comme un bloc de conception élémentaire, comme un motif présentant une certaine régularité ou fréquence dans un ensemble de données, qu'il faut savoir caractériser, extraire et ensuite interpréter. Il est possible de combiner et d'associer ces deux visions des composants et patrons : un composant — opération ou structure — devient un patron de conception s'il montre une certaine fréquence et un certain intérêt, traduisant par là qu'il se retrouve souvent employé et jugé utile (par exemple à l'aide d'une fonction de score comme pour les motifs d'intérêt maximal

dans [25]).

Quelques développements récents en ingénierie des ontologies mettent en jeu des processus de découverte de connaissances qui peuvent avoir un intérêt en matière de construction d'ontologies, en particulier à partir de documents textuels. Ces processus mettent en jeu des méthodes de fouille de données et de textes comme l'analyse formelle de concepts [10]. Par exemple, dans [5, 4], l'analyse formelle de concepts sert à extraire et formaliser des concepts à partir de tables de données binaires `objets` × `attributs` tandis que l'analyse relationnelle de concepts permet d'extraire des concepts munis d'attributs relationnels (codant des relations entre objets) à partir de données relationnelles cette fois (dans des tables `objets`×`objets`). Les treillis résultant peuvent servir de schémas d'ontologies où un concept est muni d'attributs propres et de rôles qui représentent des relations avec d'autres concepts. Toutefois, ces approches ne peuvent être entièrement automatiques pour plusieurs raisons. Un des problèmes est relatif à la couverture du corpus de textes en entrée : il est difficile de prétendre avoir toutes les connaissances explicitement disponibles pour interpréter effectivement l'ensemble des textes et souvent les connaissances sont disséminées temporellement et géographiquement. Un autre problème est qu'il est difficile d'extraire un savoir-faire et une bonne pratique à partir de documents textuels et que l'expertise est là directement nécessaire (ou encore que les bonnes pratiques sont rarement explicites mais plutôt "enfouies").

Pour en revenir au processus de conception d'ontologies lui-même, comme indiqué dans [18, 19], un utilisateur moyen essayant de mettre en place une ontologie n'a finalement que peu d'assistance disponible : des ontologies qu'il a pu trouver mais qu'il peut quelquefois très difficilement réutiliser, des constructions d'un langage de représentation de connaissances pas toujours adaptées aux besoins courants, et encore dans le meilleur des cas une brassée de bonnes pratiques glanées ça et là : par exemple les bonnes pratiques données dans [2] et `ontologydesignpatterns.org/`, qui est le site de référence en la matière et qui répertorie les composants d'ontologie disponibles jusqu'à présent. En suivant un scénario assez habituel, il aura fallu rechercher des ontologies en rapport avec le problème, les étudier et en choisir des éléments réutilisables pour les adapter. Parmi ces ontologies, certaines ontologies dites de référence sont supposées être directement réutilisables ou spécialisables comme les ontologies de haut niveau que sont DOLCE ou SUMO [24]. Là encore la tâche de compréhension est difficile et les trop nombreuses propriétés — quelquefois d'ordre métaphysique — sont rarement adaptées aux besoins tout comme la couverture de telles ontologies est bien trop grande dans la plupart des cas pratiques.

À l'opposé, il existe de petites ontologies, bien faites, concises, simples à comprendre et à utiliser (voir `ProtegeOntologiesLibrary` par exemple) mais aussi des "ontologies simplifiées" (voir encore le site `ontologydesignpatterns.org/`, avec DOLCE+DnS Ultralite par exemple). En parallèle, un certain nombre de travaux se penchent sur le problème de l'intégration d'ontologies et de la réutilisation d'ontologies par morceaux [12, 13]. Pour terminer, il faut encore mentionner des "formalismes portables" et "adaptables" qui ont acquis un certaine popularité et qui permettent de construire des ontologies avec efficacité, et en particulier : (i) SKOS [23] (pour *Simple Knowledge Organization System* qui permet de représenter des vocabulaires, des thésaurus, des partitions et des "folksonomies", et (ii) FOAF [8] qui permet de représenter des réseaux de personnes en relation.

Un parallèle avec la pratique du raisonnement à partir de cas peut être faite ici : recherche de cas similaires, évaluation des cas retrouvés puis adaptation pour réutilisation effective. L'idée est bien de comprendre comment la méthodologie du RÀPC peut aider à choisir les bons éléments dans le bon catalogue qui autorisent une adaptation minimale et efficace (ce même problème se retrouve dans la gestion des services Web [30]).

Pour aller dans ce sens, et en supposant qu'il existe des classes de problèmes qui peuvent être résolus en employant des méthodes génériques, la réutilisation d'éléments pour concevoir des ontologies doit s'appuyer sur de "petites ontologies modulaires" qui vont jouer le rôle de patrons de conception d'ontologies. Ces composants génériques particuliers doivent offrir des fonctionnalités simples et d'intérêt général et doivent être faciles d'accès et de manipulation.

3 La gestion de composants et de patrons d'ontologies

3.1 La recherche de composants et de patrons d'ontologies

La recherche d'ontologies réutilisables, de composants ou de patrons de conception d'ontologies est une tâche qui peut se faire à l'aide de moteurs spécialisés comme Sindice [31], Swoogle [14] et Watson [11, 13], par l'intermédiaire de mots-clés ou encore de "sous-structures" — ici des composants de hiérarchies de concepts. Des mesures peuvent également être utilisées pour focaliser la recherche [1, 26] :

- La mesure d'appariement de classe combine un appariement de chaînes exact et inexact et mesure le degré de correspondance entre les chaînes données en entrée et celles qui font partie des définitions de classe, pour trouver une classe qui satisfait tout ou partie des mots-clés recherchés.
- La centralité dénote une valeur qui est censée mesurer la représentativité d'un concept dans une ontologie sur la base de son placement dans la hiérarchie des concepts. Plus un concept est central en fonction de ses subsumés, de ses subsumants et de sa position par rapport au sommet de la hiérarchie des concepts, plus ce concept est "important", y compris en terme de réutilisation.
- La densité mesure le degré de détail en termes d'attributs et de rôles pour les concepts retenus. Ainsi, un concept dont la description est plus dense (donc plus riche) sera préféré à un concept de description plus minimale.
- La similarité sémantique mesure la proximité de deux concepts mis en correspondance en termes de longueur de chemins dans la hiérarchie des concepts et des relations. Ici, un appariement qui met en jeu des chemins plus courts va être préféré à un chemin plus long. Ce sont les chemins de similarité qui réapparaissent ici dont le formalisme et les opérations peuvent être repris de la même façon [22].

D'autres mesures existent aussi comme la couverture (comment un concept ou un ensemble de concepts prennent en compte les termes présents qui doivent être représentés), la structure (qui est en rapport avec la densité telle qu'elle est introduite ci-dessus), et la connectivité (qui reflète la connexion qu'un composant peut avoir avec d'autres composants dans d'autres ontologies).

Il faut faire une remarque ici, qui montre certaines limites de ces mesures : les "classes abstraites" en programmation par objets sont introduites pour factoriser des ensembles de propriétés et les distribuer entre classes. Par suite, ce sont des classes denses mais aussi centrales, toutefois elles n'ont pas de véritable existence — elles ne sont pas instanciables et instanciées — et en tout cas n'existent pas (au niveau de l'interprétation) sans les classes qu'elles alimentent.

En termes de RÀPC, la base de cas peut être considérée comme un catalogue de composants qui possèdent tous une représentation formelle et qui recouvrent de petites ontologies quelquefois réduites à un seul concept.

- L'opération de recherche de cas similaires nécessite une analyse des besoins exprimés en entrée, puis un appariement avec les cas de la base (en utilisant par exemple des mesures comme celles données ci-dessus). Les composants sélectionnés, qui répondent aux critères, tout ou partie, sont classés en fonction d'une mesure de similarité ou encore en fonction de leur adaptabilité (donc en termes de chemins de similarité comme évoqué ci-dessus).
- la recherche de composants peut aussi se voir dans ce cadre comme un alignement de composant ou plus généralement comme ce qui pourrait être qualifié "d'alignement par morceaux" comme dans [27, 28] où est définie la notion de patron de correspondances entre ontologies, qui permet d'exprimer des correspondances complexes entre parties ou morceaux d'ontologies.
- Lors de la recherche, la requête est donnée sous la forme d'une liste de mots-clés comme dans la plupart des cas, mais aussi comme une sous-structure, c'est à dire une liste de concepts vérifiant des relations de subsomption entre eux. Dans ce dernier point de vue, l'appariement est plus complexe mais va aussi retourner des cas similaires plus adaptés et adaptables.

  Cela donne en particulier des chemins de similarité "bi-dimensionnels" où une dimension fait référence à la structure (des concepts) et une autre dimension fait référence aux relations (dans lesquelles sont impliqués les concepts). Ensuite, des concepts plus généraux ou plus spécifiques peuvent être préférés selon que ce qui prime est le degré de généralité des composants ou la structure. Les concepts les plus spécifiques qui satisfont la requête vont également fournir les composants les plus spécifiques et les plus adaptables.

– Après avoir sélectionné un des composants retournés, l'étape de réutilisation consiste à spécialiser le composant et à l'adapter au problème courant, pour construire une partie d'ontologie. Une révision peut avoir lieu qui permet d'ajuster la réutilisation du composant. Une réutilisation combinée à partir de plusieurs cas peut être envisageable.
– Une opération finale de mémorisation du processus de résolution peut être entreprise si l'intérêt du composant est démontré (les mesures détaillées ci-dessus peuvent être utilisées à ce moment là). En conséquence, le composant peut se voir associer le statut de patron de conception d'ontologie.

3.2 L'adaptation de composants et patrons

Dans cette partie, il est question de composants spéciaux qui sont appelés "patrons de contenu de conception d'ontologie", pour "Content Ontology Design Patterns", abrégés ici en COP [19]. Ces composants ont une nature spéciale et permettent de traiter des problèmes de modélisation. Ils sont considérés selon un double point de vue : le domaine pour une description statique de l'environnement et la tâche pour description des opérations applicables dans l'environnement, ce qui est exactement la façon dont est perçue une classe en programmation par objets. Un même domaine peut autoriser la réalisation de plusieurs tâches tandis qu'une même tâche peut aider à résoudre un problème dans plusieurs domaines. Les ontologies quant à elles jouent plutôt un rôle au niveau du modèle de domaine mais leur conception n'implique aucune utilisation particulière : plusieurs problèmes peuvent être résolus grâce à une même ontologie.

En tant que solution à un problème de conception d'ontologie, un COP fait explicitement mention d'un domaine et d'une tâche. Une classification en catalogue de tels COPs en fonction des tâches et opérations associées peut s'avérer très utile pour fournir des scénarios standards de résolution directement ou encore en combinant les COPs. Imaginons ici la structure de graphe comme un COP qui modélise un ensemble d'entités connectées entre elles. Dans le domaine particulier des molécules en chimie organique, parmi les structures moléculaires, certaines sont génériques mais aussi d'intérêt très général et elles vont donc donner naissance à l'équivalent des COPs, les groupes fonctionnels qui décrivent des familles de réactifs. D'autres structures moléculaires sont individuelles comme les molécules particulières.

L'intuition ici est que les ontologies peuvent être construites en fonction de certaines tâches ou de certains besoins qui sont relatifs à des "questions de compétences" ("competency questions") : à quoi veut-on aboutir et quels types de problèmes est-on censé résoudre à l'aide de l'ontologie. Une question de compétence est une requête (typique) qu'un expert aurait envie de voir satisfaite à l'aide de l'ontologie. Dans l'idéal, une ontologie devrait encoder le nécessaire et pouvoir servir à satisfaire toutes les questions de compétences qui ont servi à sa conception. Dans ce cadre, un COP apparaît comme un guide de bonne pratique qui doit être décrit sous une certaine forme. Un COP a les caractéristiques d'être opérationnel, petit en taille, autonome, hiérarchique ou organisable en hiérarchie, pertinents au niveau cognitif et linguistique, et témoin d'une bonne pratique [19].

Il reste ici à développer un exemple concret et illustratif.

4 Conclusion

Cet article est une première ébauche de réflexion sur les patrons de conception d'ontologies et le raisonnement à partir de cas. Il existe un bon nombre de liens entre le deux disciplines, la conception d'ontologies et le RÀPC, dont il faut pouvoir profiter pour guider globalement la conception d'une ontologie par la recherche et l'adaptation de patrons de conception d'ontologies. En outre, une autre discipline qui est celle de la découverte de connaissances à partir de données (ressources textuelles ici) doit aussi être naturellement associée à cette étude. Comme indiqué, ceci est une première ébauche d'article qui doit encore être travaillée et approfondie.

Références

[1] H. Alani, C. Brewster, et N. Shadbolt. Ranking ontologies with aktiverank. In I.F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, et L. Aroyo, editors, *Proceedings of the 5th International Semantic Web Conference, ISWC 2006*, pages 1–15, 2006.

[2] D. Allemang et J. Hendler. *Semantic Web for the Working Ontologist*. Morgan Kaufmann, 2008.

[3] G. Antoniou et F. van Harmelen. *A Semantic Web Primer*. The MIT Press, Cambridge, Massachusetts, 2004.

[4] R. Bendaoud, A. Napoli, et Y. Toussaint. Formal concept analysis : A unified framwork for building and refining ontologies. In A. Gangemi et J. Euzenat, editors, *Knowledge Engineering : Practice and Patterns – Proceedings of the 16th International Conference EKAW*, Lecture Notes in Computer Science 5268, pages 156–171, 2008.

[5] R. Bendaoud, Y. Toussaint, et A. Napoli. Pactole : A methodology and a system for semi-automatically enriching an ontology from a collection of texts. In P. Eklund et O. Haemmerlé, editors, *Proceedings of the 16th International Conference on Conceptual Structures, ICCS 2008, Toulouse, France*, Lecture Notes in Computer Science 5113, pages 203–216, 2008.

[6] E. Blomqvist. Case-based reasoning for ontology engineering. In A. Holst, P. Kreuger, et P. Funk, editors, *Tenth Scandinavian Conference on Artificial Intelligence, SCAI 2008*, Frontiers in Artificial Intelligence and Applications 173, pages 36–43. IOS Press, 2008.

[7] E. Blomqvist. Pattern ranking for semi-automatic ontology construction. In R.L. Wainwright et H. Haddad, editors, *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008)*, pages 2248–2255. ACM, 2008.

[8] D. Brickley et L. Miller. FOAF vocabulary specification. Working draft, http ://www.foaf-project.org, 2005.

[9] M. Buffa, F.L. Gandon, G. Ereteo, P. Sander, et C. Faron. Sweetwiki : A semantic wiki. *Journal of Web Semantics*, 6(1) :84–97, 2008.

[10] P. Cimiano, A. Hotho, et S. Staab. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Journal of Artificial Intelligence Research*, 24 :305–339, 2005.

[11] M. d'Aquin, C. Baldassarre, L. Gridinoc, M. Sabou, S. Angeletou, et E. Motta. Watson : Supporting next generation semantic web applications. In *Proceedings of WWW/Internet Conference*, 2007.

[12] M. d'Aquin, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez, et D. Guidi. Toward a new generation of semantic web applications. *IEEE Intelligent Systems*, 23(3) :20–28, 2008.

[13] M. d'Aquin, M. Sabou, et E. Motta. Reusing knowledge from the semantic web with the watson plugin. In *International Semantic Web Conference (Session Posters & Demos)*, CEUR Workshop Proceedings 401, 2008.

[14] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, et P. Kolari. Finding and ranking knowledge on the semantic web. In Y. Gil, E. Motta, V. R. Benjamins, et M. A. Musen, editors, *The Semantic Web - ISWC 2005 : 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland*, Lecture Notes in Computer Science 3729, 2005.

[15] J. Euzenat et P. Shvaiko. *Ontology Matching*. Springer, Berlin, 2007.

[16] D. Fensel, J. Hendler, H. Lieberman, et W. Wahlster, editors. *Spinning the Semantic Web*. The MIT Press, Cambridge, Massachusetts, 2003.

[17] E. Gamma, R. Helm, R. Johnson, et J. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading (MA), USA, 1994.

[18] A. Gangemi. Ontology design patterns for semantic web content. In Y. Gil, E. Motta, V. R. Benjamins, et M. A. Musen, editors, *The Semantic Web - ISWC 2005 : 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland*, Lecture Notes in Computer Science 3729, 2005.

[19] A. Gangemi et V. Presutti. Ontology Design Patterns. In S. Staab et R. Studer, editors, *Handbook on Ontologies (Second Edition)*. Springer, Berlin, 2008.

[20] K. Hua, B. Faltings, et I. Smith. CADRE : case-based geometric design. *Artificial Intelligence in Engineering*, 10 :171–183, 1996.

[21] M. Krötzsch, D. Vrandecic, et M. Völkel. Semantic mediawiki. In *The Semantic Web – ISWC 2006*, Lecture Notes in Computer Science 4273, pages 935–942. Springer, 2006.

[22] J. Lieber et A. Napoli. Correct and Complete Retrieval for Case-Based Problem-Solving. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98), Brighton, UK*, pages 68–72. John Wiley & Sons Ltd, Chichester, 1998.

[23] A. Miles et D. Brickley. SKOS Core Vocabulary Specification. Technical report, World Wide Web Consortium (W3C), 2005.

[24] D. Oberle, A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel, B. Mougouie, S. Baumann, S. Vembu, et M. Romanelli. DOLCE ergo SUMO : On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *Journal of Web Semantics*, 5 :156–174, 2007.

[25] F. Pennerath, G. Polaillon, et A. Napoli. Mining intervals of graphs to extract characteristic reaction patterns. In J.-F. Boulicaut, M.R. Berthod, et T. Horváth, editors, *Discovery Science*, Lecture Notes in Computer Science 5255, pages 210–221. Springer, Berlin, 2008.

[26] S. Peroni, E. Motta, et M. d'Aquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In J. Domingue et C. Anutariya, editors, *Proceedings of the 3rd Asian Semantic Web Conference, ASWC 2008*, Lecture Notes in Computer Science 5367, pages 242–256. Springer, 2008.

[27] F. Scharffe, J. Euzenat, et D. Fensel. Towards design patterns for ontology alignment. In R.L. Wainwright et H. Haddad, editors, *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008)*, pages 2321–2325. ACM, 2008.

[28] F. Scharffe et D. Fensel. Correspondence patterns for ontology alignment. In A. Gangemi et J. Euzenat, editors, *Knowledge Engineering : Practice and Patterns – Proceedings of the 16th International Conference EKAW*, Lecture Notes in Computer Science 5268, pages 83–92, 2008.

[29] S. Staab et R. Studer, editors. *Handbook on Ontologies*. Springer, Berlin, 2004.

[30] R. Studer, S. Grimm, et A. Abecker, editors. *Semantic Web Services – Concepts, Technologies and Applications*. Springer, Berlin, 2007.

[31] G. Tummarello, R. Delbru, et E. Oren. Sindice.com : Weaving the open linked data. In K. Aberer, K.-S. Choi, N. Fridman Noy, D. Allemang, K.-I. Lee, L.J.B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, et P. Cudré-Mauroux, editors, *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, ISWC/ASWC 2007*, Lecture Notes in Computer Science 4825, pages 552–565. Springer, 2007.

Conception d'un Système de Diagnostic Industriel par Raisonnement à Partir de Cas

Karim Haouchine, Brigitte Chebel-Morello et Noureddine Zerhouni
Institut FEMTO-ST, UMR CNRS 6174 - UFC / ENSMM / UTBM,
Département d'Automatique et Systèmes Micro-Mécatroniques (AS2M),
24, rue Alain Savary, 25000 Besançon, France.
{karim.haouchine, bmorello, zerhouni}@ens2m.fr

Résumé

Ce papier décrit une démarche de conception d'un système de diagnostic industriel par RàPC. C'est un système orienté connaissance. La formalisation du cas prend appui sur la définition du diagnostic. Le cas a une formalisation orientée objet qui est issue d'une analyse dysfonctionnelle de l'équipement. Nous avons associé à ces cas deux modèles représentant les connaissances de l'équipement à diagnostiquer. Le premier modèle est issu d'une analyse fonctionnelle regroupant les composants qui assurent les mêmes fonctions. Le deuxième modèle repose d'une part sur le flux des entités de l'équipement et sur l'analyse de sa décomposition et d'autre part, sur les relations de cause-à-effet entre les composants. Toutes les phases du cycle de RàPC dépendent de la formalisation des cas associée aux modèles de connaissance et plus spécifiquement les phases de remémoration et d'adaptation. Ces deux phases sont liées entre elles. Nous avons ainsi proposé une méthode de remémoration guidée par l'adaptation en mettant en place deux mesures, à savoir : la mesure de remémoration et la mesure d'adaptation. Ces mesures permettent de choisir le cas le plus facilement adaptable lors de la phase de remémoration. Ensuite, nous avons mis en place un algorithme d'adaptation qui prend appui sur les relations de dépendance des descripteurs de problème et de solution. Une étude de la faisabilité de notre algorithme est faite sur un moteur à explosion de type 1.5 dCi K9K 105ch de la société Renault.

Mots clés : Raisonnement à partir de cas, diagnostic, modèles de connaissance, remémoration guidée par l'adaptation, relations de dépendance

1 Introduction

Plusieurs systèmes et applications de raisonnement à partir de cas (RàPC) ont été construits ces dernières années. De plus, il existe plusieurs états de l'art dans ce domaine [23], [3] dans différentes applications (médecine, ingénierie, juridique, cuisine), et dans divers types de problèmes (diagnostic, planification, conception, help desk ....). Notre étude se focalise sur les systèmes complexes de diagnostic industriel. Il existe une large variété de méthodes dans ce domaine allant des systèmes conversationnels tel que le système « checkmate » [9], relatif aux imprimantes industrielles par Domino UK Ltd., jusqu'aux systèmes de type « help desk » tels que CaseLine [23], qui est un démonstrateur utilisé par British Airways pour le diagnostic de pannes et la réparation de Boeing 747-400, et CASSIOPEE [Bergmann et al., 2003] qui s'occupe de dépannage de moteurs d'avions Boeing 737 à CFM International. Nous pouvons citer d'autres systèmes de diagnostic industriel tel que le système IRACUS [Varma, 1999] qui est dédié au diagnostic de pannes des locomotives utilisé par les systèmes « GE Transport » et FormTool qui définit et produit les couleurs du plastique désirées par le client [4]. Les deux derniers systèmes ne disposent pas de modèles de connaissance comme le système FormTool Gas Turbine qui est exploité par General Electric Energy [7]. Nous pouvons encore citer le système NodalCBR [6] qui exploite un modèle de classification sous forme d'arbre. Ce système est dédié au diagnostic de pannes des alimentations d'énergie de mode de commutation (SMPS). Par ailleurs, certains systèmes de diagnostic ont un outil de représentation des connaissances spécifique, tels que CREEK [1] qui est un système de diagnostic de voitures et qui dispose de l'outil $CREEK_1$, Pad'im [15] qui développe la conception des systèmes de supervision en industrie et qui dispose de

l'outil Rocade, Patdex [18] qui est un système de diagnostic technique pour des machines complexes disposant de l'outil Moltke et Maic [16] qui est un système de maintenance des équipements chimiques qui dispose de l'outil Optrans.

Toutefois, d'après cette étude et suivant la diversité des méthodes présentées, on serait amené à penser que ces méthodes sont spécifiques à l'application, et qu'il n'y a pas de méthode générale de conception d'un système de diagnostic par RàPC. De plus, dans la plupart des systèmes, le cas caractérise une expérience de diagnostic sans faire référence à un type de modèle.

Dans ce papier, nous désirons définir une méthodologie de conception d'un système d'aide au diagnostic, problématique que nous avons initiée par nos travaux précédents dans [17]. Nous avons développé une méthode de management des connaissances pour modéliser l'expertise en diagnostic en lien avec les outils de sureté de fonctionnement. Cependant, dans ce papier, on partira de la définition du diagnostic pour mettre en place la modélisation de cas. L'élément de base dans notre démarche est l'exploitation des outils de sureté de fonctionnement, en lien avec l'AMDEC et les arbres de défaillances de l'équipement à étudier.

Un des points clés du RàPC réside dans ses phases de remémoration et d'adaptation. La majorité des systèmes étudiés ne disposent pas de phase d'adaptation. Nous développons à la section 2 les travaux majeurs qui ont été faits sur la relation entre ces deux phases et plus spécifiquement sur la remémoration guidée par l'adaptation (AGR). Nous présentons à la section 3 la méthodologie de conception d'un système de diagnostic par RàPC orienté connaissance. Cette conception passe par la mise en place des modèles de connaissance de l'équipement à diagnostiquer. Ces modèles sont issus d'une analyse fonctionnelle, d'une analyse de l'historique de pannes et de la décomposition de cet équipement. Nous associons à ces modèles l'expertise terrain qui reflète l'analyse dysfonctionnelle des composants de l'équipement sous forme de cas dans la base de cas. La formalisation du cas prend appui sur la définition du diagnostic. Par ailleurs, nous proposons dans la même section deux mesures complémentaires. Ces mesures permettant de retrouver les cas les plus similaires, quand un problème à résoudre se présente, et de sélectionner le cas le mieux adaptable. Ces mesures sont intimement liées à la modélisation du cas. Nous agrémentons chaque étape de cet article par un exemple d'un moteur diesel de type 1.5 dCi K9K 105ch de la société Renault, disponible à l'adresse suivante : http://v3.renault.com/cfm/module-K9K/fr/index.html. Nous illustrons à la section 4 les phases de remémoration et d'adaptation à travers trois exemples types de diagnostic. Enfin, nous terminons, à la section 5, par une évaluation des méthodes de remémoration et d'adaptation mises en place.

2 Remémoration guidée par l'adaptation

Avant les années 90, les deux phases de remémoration et d'adaptation étaient exploitées d'une façon complètement indépendante. Jusqu'à ce que Smyth et Keane [21] apportent un nouveau souffle et suggèrent l'unification de ces deux étapes. Cette unification va dans le sens où le cas choisi dans l'étape de remémoration est celui le plus facilement adaptable afin d'optimiser les résultats de l'étape d'adaptation. Dans ces travaux d'unification, la mesure de similarité est combinée avec d'autres critères afin de guider le processus de remémoration. Lopez de Mantaras a proposé six types de remémoration liée à l'adaptation [14]. Nous nous intéressons particulièrement à la remémoration guidée par l'adaptation. Ces travaux ont été initiés par Smyth et Keane [21] et sont connus sous le nom de : *Adaptation-Guided Retrieval*, abrégé : « AGR ». Ce type de remémoration a été le point de départ de plusieurs travaux prometteurs. Les mêmes travaux ont été présentés ensuite dans [20] et [22] qui donnent l'argument que ce n'est pas le cas le plus similaire, quand la mesure de similarité est choisie a priori, qui est le meilleur candidat à l'adaptation. De ce fait, la phase de remémoration doit donc rechercher non seulement des cas similaires pendant son processus mais surtout des cas facilement adaptables. En effet, les auteurs expliquent comment cette remémoration qui est guidée par l'adaptation lie les espaces des spécifications et des solutions en employant la connaissance d'adaptation. Ce lien permet d'avoir un canal de communication entre la remémoration et l'adaptation. Par conséquent, pendant la phase de remémoration, les connaissances d'adaptation sont utilisées pour intervenir en cas d'éventuels changements dans la solution.

Collins et Cunningham [5] et Rousu et al. [19] abordent le principe du calcul de *l'effort d'adaptation* dans des problèmes de planification.

Dans le même registre, Leake et ses collègues [12] abordent également la notion de l'effort d'adaptation et la répercussion des traditionnelles mesures de similarité sémantiques sur l'adaptation. Certes, les cas

remémorés sont « similaires » au problème cible, mais parfois difficiles voire impossible à adapter. Les auteurs prennent en compte l'effort d'adaptation au moment de la remémoration afin de faciliter la phase d'adaptation. Cette prise en compte est concrétisée par l'insertion du coût d'adaptation dans la mesure de similarité. Alors, la méthode proposée s'appuie sur deux étapes lors de l'évaluation de la similarité entre les cas sources de la base de cas et le problème cible : une première étape de remémoration suivie par une étape d'un ordonnancement des cas remémorés en fonction d'un coût d'adaptation.

Par ailleurs, Lieber [13] propose une approche d'adaptation s'appuyant sur la notion de chemins de similarité. Cette notion est basée sur l'idée de décomposition de l'adaptation en sous tâches d'adaptation plus simples. Pour cette décomposition, il faut disposer de connaissances dépendantes du domaine. Cette approche comporte deux étapes. La première étape consiste à construire le chemin de similarité. La deuxième étape consiste à calculer les petites adaptations, qui se feront dans l'étape d'adaptation.

3 Le système de diagnostic par RàPC

Une première étude a été faite dans notre équipe concernant la conception d'un système de RàPC dédié au diagnostic [17]. Le cas est formalisé sous forme d'une représentation d'objet pour les avantages qu'elle amène. En effet, cette représentation permet d'élaborer aisément une structure hiérarchique de descripteurs qui pourrait être exploitée dans les phases de remémoration et d'adaptation. Cette étude commence par la capitalisation des connaissances d'un opérateur lors d'un dépannage. Ces connaissances sont proposés aux opérateurs afin d'améliorer la performance et l'efficacité de leurs interventions au niveau du service de maintenance. Une modélisation de l'expertise est proposée. Celle-ci est issue des outils d'analyse par des méthodes de représentation et d'une méthode de modélisation des connaissances qui est issue du domaine d'ingénierie des connaissances.

La mise en œuvre de notre système de RàPC repose tout d'abord sur l'acquisition et la représentation des connaissances des différentes données. La représentation des connaissances se fait sur la base de deux modèles sous-jacents, à savoir : le modèle de contexte et le modèle de taxonomie des composants.

L'analyse de l'équipement détermine des ensembles de ces composants. Chaque ensemble regroupe les composants qui assurent les mêmes fonctions. Cette analyse fonctionnelle permet donc d'avoir une modélisation hiérarchique de l'équipement. La figure 1 montre un exemple du modèle hiérarchique des composants d'un moteur à explosion. En effet, ce moteur va servir d'exemple d'illustration tout au long de l'article.

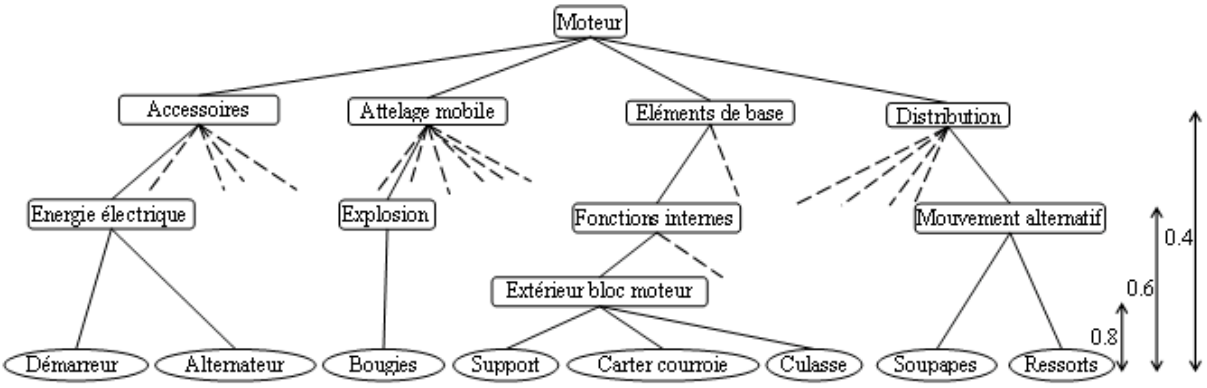


Fig. 1. Modèle hiérarchique des composants d'un moteur à explosion.

Comme le moteur dispose d'une multitude de composants, nous montrons qu'une partie du modèle hiérarchique des composants. Ce modèle comporte quatre classes : Attelage mobile, Distribution, Eléments de base et Accessoires. Chaque classe est composée de sous-classes jusqu'à ce qu'on arrive au niveau des composants.

Le modèle de contexte se base sur les flux des entités du système et sur l'analyse de la décomposition de l'équipement du système étudié. Cette décomposition détermine les fonctions assurées par l'équipement et ses composants. Ces composants constituent donc le contexte dans lequel le composant défaillant est identifié. Le contexte reflète les relations de cause-à-effet permettant d'une part de localiser des composants à problèmes et d'autre part, de sélectionner les bons descripteurs par rapport à l'ensemble. La figure 2 montre un aperçu du modèle de contexte d'un moteur à explosion.

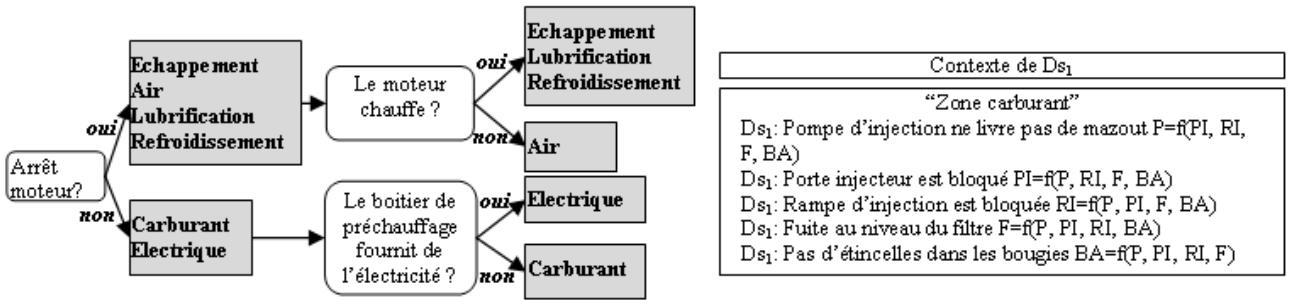


Fig. 2. Aperçu du modèle de contexte d''un moteur à explosion.

Le modèle de contexte est mis en place selon les différentes natures du flux dans le moteur. Les flux que nous avons identifiés sont : échappement, air, lubrification, refroidissement, carburant et électrique. Chaque flux implique un certain nombre de composants. Chaque classe peut contenir plusieurs flux et également chaque flux peut contenir plusieurs classes.

Ces deux modèles sont reliés à une base de cas qui reflète la partie dysfonctionnelle des composants. Les cas dans la base de cas ont une représentation orientée objet qui est issue d'une analyse de l'historique des pannes, des arbres de défaillances des composants de l'équipement ainsi que de l'étude de l'AMDEC. Cette représentation prend appui sur la définition de référence du diagnostic qui est la suivante : « *ce sont les actions menées pour la* ***détection*** *de la panne, sa* ***localisation*** *et* ***l'identification*** *de la cause* [2] ». Nous exploitons ces trois parties dans la formalisation du cas, qui de plus, prendra appui sur les modèles de connaissances de l'équipement à diagnostiquer. Nous aurons ainsi la partie localisation et la partie fonctionnelle dans l'espace problème du cas et, la partie détection de la classe de défaillance et l'identification du composant défaillant dans l'espace solution du cas. Une des spécificités de notre cas d'étude est de déterminer un état et un mode de fonctionnement à chaque composant à diagnostiquer [11]. On affectera par conséquent à un descripteur donné :

- Un attribut relatif à sa valeur proprement dite (le descripteur est un composant électrique ou un composant mécanique par exemple) ds_i^{Valeur} ;
- Un attribut relatif à l'état de celui-ci ds_i^{Etat} ;
- Ainsi qu'un attribut relatif au mode de fonctionnement qui reflétera l'état normal et/ou anormal des composants de l'équipement $ds_i^{M.F}$.

Les descripteurs de problème de la partie fonctionnelle auront par conséquent trois attributs relatifs à la valeur du composant, son état et son mode de fonctionnement : dsi = (ds_i^{Valeur} , ds_i^{Etat} , $ds_i^{M.F}$).

Par conséquent, la formalisation du cas aura une structure qui est montrée sur la figure 3.

Partie problème								Partie solution		
Localisation	Fonctionnelle							Classe	Composant défaillant	..
$ds_1 \ldots ds_l$	ds_{l+1}^{valeur}	ds_{l+1}^{Etat}	$ds_{l+1}^{M.F}$	…	ds_{m+1}^{valeur}	ds_{m+1}^{Etat}	$ds_{m+1}^{M.F}$	Ds_1	Ds_2	..

Fig. 3. Structure générique du cas

Les figures 4 et 5 montrent un aperçu des parties problème et solutions de 7 cas de la base de cas d'un moteur à explosion.

	Problème																												
	localisation				Partie fonctionnelle																								
	ds1	ds2	ds3	ds4	ds5			ds6			ds7			ds8			ds9			ds10			ds11			ds12		
Index	Etat moteur	Bougies de préchauffage	T° moteur	Sous-zone	Injection	Etat	M.F	Filtrage	Etat	M.F	Explosion	Etat	M.F	Transition	Etat	M.F	Pression	Etat	M.F	Accumulation	Etat	M.F	Mouvement par frottement	Etat	M.F	Mouvement alternatif et rotatif	Etat	M.F
1 (RD forte)	Tourne	Mauvais	-	-	Pompe injection	Non entraî-née	A	-	-	-	Bougies	Etin-celle	N	-	-	-	Filtre	Gaz circ-ule	N	-	-	-	-	-	-	Vilebrequin	Mouv. contin u	N
2 (RD faible)	Arrêt	-	Haute	Partie haute	-	-	-	Pot catalyt-ique	Etan-che	N	-	-	-	Refroidi-sseur	Circ-ule l'air	N	-	-	-	R.G.E	Bonne press-ion	N	-	-	-	-	-	-
3 (RD forte)	Tourne	Mauvais	-	-	Porte injecteur	Transit partie-lle	A	-	-	-	Bougies	Etin-celle	N	-	-	-	Electro-vanne	Carbu passe pas	A	-	-	-	-	-	-	-	-	-
4 (RD forte) & ≠	Tourne	-	-	-	Pompe injection	Entraî-née	N	Monoli-the	Air suffi-sant	N	Bougies	Etin-celle	N	-	-	-	-	-	-	-	-	-	Arbre à cames	Mouv. disc-ontinu	A	-	-	-
5 (RD forte) & ≠	Arrêt	-	Moye-nne	-	-	-	-	Pot catalyt-ique	Etan-che	N	Bougies	Etin-celle	N	Joint de collecteur	Etan-che	N	-	-	-	Vanne de recirculation	Pass-age d'air	N	Poussoirs	Mouv. dévié axe	A	DVA	Mouv. contin u	N
6 (RD faible)	Tourne	Mauvais	-	-	Rampe injection	Absе-nce parti-cule	N	-	-	-	Bougies	Etin-celle	N	-	-	-	Turbo-compre-sseur	aucun bruit	N	-	-	-	Courroie de distribution	Serré e	N	-		-
7 (RD Forte)	Tourne	Mauvais	-	-	Pompe injection	Entraî-née	N	-	-	-	Bougies	Etin-celle	N	-	-	-	Compres seur	Gaz circ-ule	N	-	-	-	-	-	-	Vilebrequin	Mouv. disc-ontinu	A

Fig. 4. Aperçu de la partie problème de la base de cas d'un moteur à explosion.

	Solution			
	Ds1	Ds2	Ds3	Ds4
Index	Classe de défaillance	Identification composant défaillant	Action de réparation associée	Zone de défaillance
1 (RD forte)	Attelage mobile	Pompe injection "livre pas mazout"	Changer pompe	Carburant
2 (RD faible)	Attelage mobile	Turbo-compresseur "Prise d'air"	Changer filtre	Echappe-ment
3 (RD forte)	Attelage mobile	Electro-vanne "pbm masse"	Replacer files électro-vanne	Carburant
4 (RD forte) & ≠	Mouvement par pression	Pistons "déformés"	Changer pistons	Carburant + Echappe-ment
5 (RD forte) & ≠	Mouvement par pression	Axes pistons "décalés"	Replacer axes pistons	Carburant + Echappe-ment
6 (RD faible)	Mouvement rotatif	Pignon vilebrequin "bloqué"	Replacer pignon vilebrequin	Carburant + Echappe-ment
7 (RD Forte)	Mouvement rotatif	Vilebrequin "décalé"	Lubrifier vilebrequin	Carburant

Fig. 5. Aperçu de la partie solution de la base de cas d'un moteur à explosion.

Prenons l'exemple du cas 1. Ce dernier reflète une défaillance au niveau de la pompe d'injection. La partie problème détermine dans sa partie « localisation » l'endroit du composant défaillant qui est défini par le modèle de contexte. D'après les quatre premiers descripteurs, le moteur tourne, les bougies de préchauffage sont dans un mauvais état et les deux descripteurs restants ne sont pas renseignés. Lorsqu'on se réfère au modèle de contexte qui est présenté précédemment, nous trouvons un certain nombre de composants qui sont potentiellement défaillants dans la zone spécifiée. Ensuite, la partie fonctionnelle du cas indique l'état des composants dans cette zone. Le descripteur « ds_5 » qui reflète le composant « pompe d'injection » appartenant à la classe « injection » est dans un état « non entraîné » engendrant un mode de fonctionnement « anormal ». Le descripteur « ds_7 » reflète l'état de la bougie qui est dans un mode de fonctionnement « normal » car elle produit des étincelles. Le descripteur « ds_9 » décrit le composant filtre qui fait circuler normalement le gaz. Le descripteur « ds_{12} » indique que le vilebrequin a un mouvement continu et est en mode de fonctionnement « normal ». Les autres descripteurs ne sont pas renseignés.

Quant à la partie solution, le descripteur « Ds_1 » indique que la classe de défaillance du composant concerné est « Attelage mobile ». « Ds_2 » précise que le composant défaillant est la « pompe d'injection » accompagné d'une remarque que cette pompe ne livre pas de mazout. Ainsi, l'action de réparation associée qui est décrite par le descripteur « Ds_3 » préconise un changement de la pompe d'injection. Enfin, le descripteur « Ds_4 » indique que la défaillance s'est produite au niveau du flux de passage du carburant.

La phase de remémoration tiendra compte de trois types d'attributs de la partie fonctionnelle du cas. La phase d'adaptation exploitera les relations de dépendance en parcourant un modèle de contexte qui définit les relations de cause-à-effet entre les descripteurs.

3.1 Phase de remémoration

Afin de sélectionner, parmi les cas remémorés, le cas le plus favorable à l'adaptation, nous devons dans un premier temps évaluer la ressemblance des cas sources avec le cas cible en appliquant une mesure de similarité globale qui est composée par un ensemble de mesures de similarité locales [11]. Ensuite, nous tiendrons compte des modes de fonctionnement des descripteurs de la partie fonctionnelle via un poids λi dans une deuxième mesure qu'on nommera : *mesure d'adaptation (M_A)*.

3.1.1 Mesure de remémoration

Nous avons initié la mesure de remémoration (M_R) dans [11]. Nous rappelons que cette mesure dépend de quatre mesures de similarité locales, à savoir : une mesure qui dépend de la classe d'appartenance des valeurs des descripteurs φ^{Valeur} (elle prend appui sur les valeurs des niveaux hiérarchiques du modèle de taxonomie des composants comme le montre la figure 1), une mesure qui dépend de l'état du descripteur φ^{Etat}, une mesure qui dépend de la présence des valeurs des descripteurs $\varphi^{\text{Pr}ésence}$ et une mesure qui dépend du mode de fonctionnement $\varphi^{M.F}$.

Ainsi la mesure de remémoration (M_R) représente l'agrégation de ces quatre mesures de similarité locales et se traduit par la formule suivante :

$$M_R = \frac{\sum_{i=1}^{m} \varphi_i^{Valeur} \times \varphi_i^{Etat} \times \varphi_i^{\text{Pr}ésence} \times \varphi_i^{M.F}}{\sum_{i=1}^{m} \varphi_i^{\text{Pr}ésence}} \qquad (1)$$

Où *m* est le nombre de descripteurs de problème.

Cette mesure va donc fournir un ensemble de cas les plus similaires au cas cible. Afin de sélectionner le cas source remémoré le plus adaptable, nous avons mis en place une mesure appelée : mesure d'adaptation (M_A).

3.1.2 Mesure d'adaptation

La Mesure d'Adaptation (M_A) tient compte du mode de fonctionnement des composants en accordant de l'importance aux modes de fonctionnement anormaux dans les descripteurs du cas source (dsi) et du cas cible (dci), en leur affectant un poids plus important. Cette importance sera caractérisée par le poids λi. Ce poids est primordial dans la détermination du composant défaillant. La présence de la valeur du descripteur est prise en compte, ce qui permet la facilité d'adaptation. Enfin, la valeur du descripteur est également prise en compte car plus la valeur du descripteur cible est proche de la classe de la valeur du descripteur source et plus ce descripteur est facilement adaptable.

De ce fait, la Mesure d'Adaptation (M_A) est définie comme suit :

$$M_A = \frac{\sum_{i=1}^{m} \lambda_i . \times \varphi_i^{Présence} \times \varphi_i^{Valeur}}{\sum_{i=1}^{m} \varphi_i^{Présence}} \qquad (2)$$

Où λi représente le poids associé en fonction du mode de fonctionnement.

$$\begin{cases} \text{Si MF(dsi, dci) = \{normal/normal\}} \rightarrow \lambda i = 2^0 \\ \text{Si MF(dsi, dci) = \{anormal/normal ou normal/anormal\}} \rightarrow \lambda i = 2^1 \\ \text{Si MF(dsi, dci) = \{anormal/anormal\}} \rightarrow \lambda i = 2^2 \end{cases}$$

Les descripteurs qui sont en mode défaillant sont privilégiés en imposant le double du poids choisi car ils représentent les composants qui sont les plus susceptibles d'être défaillants dans l'espace solution. Par conséquent, le cas source possédant la plus grande valeur de la mesure d'adaptation parmi les cas sources remémorés sera le candidat choisi pour la phase d'adaptation.

3.2 Phase d'adaptation

La phase d'adaptation est réalisée grâce à un algorithme d'adaptation. Cet algorithme prend appui sur le modèle de contexte, sur le modèle hiérarchique des descripteurs et sur les relations de dépendance entre les différents descripteurs de problème et de solution. Les relations de dépendance sont inspirées des travaux de Fuchs dans [8]. Ces relations permettent d'exprimer l'influence d'un descripteur problème « ds » sur les descripteurs solution « Ds » dans les différents cas sources de la base de cas. Ces relations vont aider à définir un ensemble pertinent de descripteurs qui pourront être utilisés lors de l'étape d'adaptation. Leur mise en place est expliquée dans [10]. Trois valeurs sont attribuées aux différents descripteurs des cas source suivant le lien entre l'espace problème et l'espace solution, $RD_{ij} \subset$ {forte ; faible ; pas de relation}.

L'algorithme d'adaptation est présenté dans [11]. Cet algorithme d'adaptation adapte descripteur par descripteur. Nous avons identifié trois cas type d'adaptation à savoir :

- ***RD forte*** impliquant au moins un descripteur de solution avec un descripteur de problème appartenant à la ***même classe de fonctionnement*** ;
- ***RD forte*** impliquant au moins un descripteur de solution avec un descripteur de problème appartenant à des ***classes de fonctionnement différentes*** ;
- ***RD faible***, lorsque les composants représentés par les descripteurs de la partie problème ont une répercussion indirecte sur les descripteurs de la partie solution.

Nous pouvons voir sur la figure 4 que l'index du cas 1 montre que ce cas a une relation de dépendance forte qui implique un composant dans la partie problème appartenant à la même famille que celle du composant indiqué dans la partie solution.

4 Illustration des phases de remémoration et d'adaptation

L'étude de la gestion des connaissances, la formalisation du cas, la méthode de remémoration guidée par l'adaptation proposée ainsi que l'algorithme d'adaptation sont appliqués sur un moteur à explosion 1.5 dCi K9K 105ch de la société Renault disponible à l'adresse suivante : http://v3.renault.com/cfm/module-K9K/fr/index.html.

Une étude fonctionnelle et dysfonctionnelle de ce moteur a été faite dans l'optique de la mise en place complète de notre méthode et la vérification de la faisabilité des résultats obtenus. Grâce à cette étude, nous avons construit une base de cas contenant 20 cas. Un cas, dans sa partie problème est composé de 12 descripteurs. Les quatre premiers descripteurs déterminent la zone de défaillance de l'équipement « moteur ». Les huit descripteurs restants : [ds_5,…, ds_{12}], déterminent la partie fonctionnelle des composants appartenant à la zone de défaillance de l'équipement « moteur ». Les descripteurs de cette partie représentent les classes de fonctionnement des différents composants de l'équipement qui se trouvent dans la hiérarchie

des composants.

Concernant la partie solution du cas, qui est composée de quatre descripteurs, nous trouvons la classe de défaillance du composant concerné par les symptômes décrits dans la partie problème, bien évidemment le composant défaillant, les actions à mener pour remédier à cette panne (ou défaillance) et l'endroit dans lequel il faut intervenir.

4.1 Premier exemple type d'adaptation : « RD = Forte & même classe de fonctionnement »

Soit une panne produite dans le moteur au niveau des *coussinets de vilebrequin* et représentée par le cas *cible 1* (figure 6).

	ds1	ds2	ds3	ds4	ds5			ds6			ds7			ds8			ds9			ds10			ds11			ds12		
Cible1	Tourne	Mauvais			Pompe injection	Non entraî-née	A				Bougies	Etin-celle	N				Filtre	Gaz circ-ule	N							Coussinets vilebrequin	Surf-ace rugu-euse	A
1 (RD forte)	Tourne	Mauvais			Pompe injection	Non entraî-née	A				Bougies	Etin-celle	N				Filtre	Gaz circ-ule	N							Vilebrequin	Mouv. cont-inu	N

Fig. 6. Le cas source le plus adaptable au cas cible 1.

Nous précisons que nous avons fixé un seuil de similarité de 60% dans la phase de remémoration.

Maintenant, nous procédons au calcul de la mesure de remémoration (M_R). Les cas sources les plus similaires au cas cible 1 sont :

$$M_R(srce_1, cible_1) = \frac{(1\times 1) + (1\times 1) + (1\times 1\times 1\times 1) + (1\times 1\times 1\times 1) + (1\times 1\times 1\times 1) + (1\times 0\times 0.8\times 0)}{6}$$

$\varphi_i^{Présence}$ φ_i^{Etat} φ_i^{Valur} $\varphi_i^{M.F}$

ds_1 ds_2 ds_5 ds_7 ds_9 ds_{12}

$M_R(srce_1, cible_1) = \mathbf{0.83}$

$$M_R(srce_4, cible_1) = \frac{(1\times 1) + (1\times 0\times 1\times 0) + (1\times 1\times 1\times 1)}{3} = 0.67$$

Les deux cas sources les plus similaires au cas cible 1 sont donc les cas 1 et 4.

Maintenant, nous procédons au calcul de la mesure d'adaptation (M_A) des deux cas sources par rapport au cas cible 1. Nous rappelons que la mesure M_A est calculée seulement par rapport aux descripteurs de la partie fonctionnelle du cas en l'occurrence du descripteur ds_5 au descripteur ds_{12}.

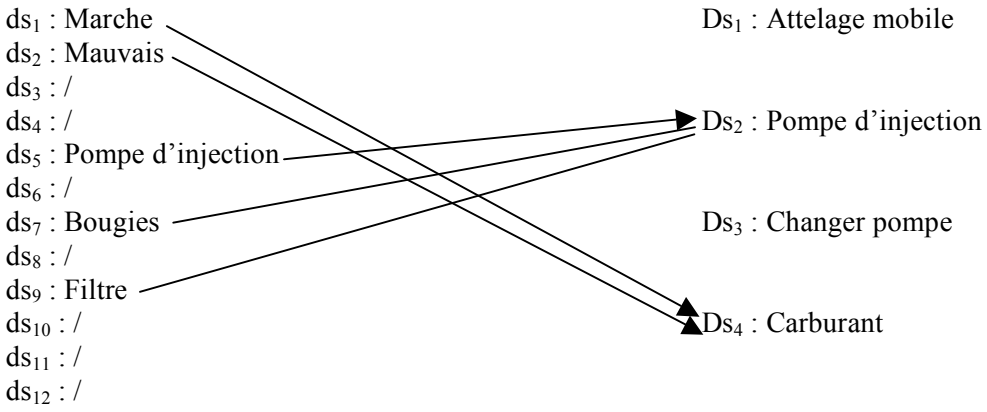
$$M_A(srce_1, cible_1) = \frac{(4\times 1\times 1) + (1\times 1\times 1) + (1\times 1\times 1) + (2\times 1\times 1)}{4} = \mathbf{2}$$

$M_A(srce_4, cible_1) = 1.5$

$M_A(srce_1, cible_1) > M_A(srce_4, cible_1)$. Par conséquent, le cas source 1 est le plus facilement adaptable au cas cible 1.

Nous constatons que dans cet exemple le cas source le plus similaire au cas cible 1 est également celui le plus facilement adaptable suivant sa valeur de M_A.

La représentation des relations de dépendance entre les descripteurs de problème et les descripteurs de solution du cas source 1 est la suivante :



Nous rappelons que les flèches représentent les « RD = forte », les segments « RD = faible » et lorsqu'il n'y a rien qui relie les descripteurs entre eux alors « RD = pas de relation ».

Dans l'étape d'adaptation, les quatre premiers descripteurs sont destinés à déterminer la zone de défaillance qui est exprimée par le descripteur Ds_4. Les descripteurs « ds_5… ds_{12} » concernent la détermination du composant défaillant et sont liés directement au descripteur Ds_2. Les descripteurs Ds_1 et Ds3 sont la conséquence de la valeur du descripteur Ds_2.

Dans un premier temps, nous remarquons que la valeur de RD du couple (Ds_2, ds_5) est « RD = Forte ». Ensuite, les deux descripteurs Ds_2 et ds_5 représentent le même composant « pompe d'injection » et donc ils ont la même classe d'appartenance. De ce fait, nous sommes dans le cas de figure de « RD = Forte et même classe de fonctionnement ».

En appliquant l'algorithme d'adaptation, nous obtenons :

- Substituer la valeur du descripteur ds_5^{rem} qui est en mode « anormal » par la valeur du descripteur dc_3 = pompe d'injection ;
- Les deux valeurs ds_5^{rem} et dc_3 sont égales alors on affecte directement la solution de Ds_2 à la solution cible Dc_2 : Dc_2 = Pompe d'injection « ne livre pas de mazout ».

La solution est donc la suivante : *changer la pompe d'injection qui ne livre pas le mazout se trouvant dans la zone « carburant »*.

4.2 Deuxième cas type d'adaptation : « RD = Forte & classes de fonctionnement différentes »

Soit une panne qui s'est produite dans le moteur au niveau du *joint de collecteur* et représentée par le cas *cible 2* (figure 7).

	d1	d2	d3	d4	d5			d6			d7			d8			d9			d10			d11			d12		
Cible2	Arrêt		Moye-nne					Monoli-the	Air insu-ffisan t	A	Bougies	Etin-celle	N	Joint de collecteur	Etan-che	N				R.G.E	Bonn e press-ion	N	Arbre à cames	Mouv. Conti nu	A	DVA	Mouv. cont-inu	N
16 (RD forte) & ≠	Arrêt		Haute	Par-tie ba-sse	Pompe injection	Entraî-née	N	Monoli-the	Air suffi-sant	N	Bougies	Etin-celle	N				Turbo-compre-sseur	auc-un bruit	N				Arbre à cames	Mouv. Conti nu	A			

Fig. 7. Le cas source le plus adaptable au cas cible 2.

- ***Calcul de la mesure de remémoration***

$M_R(srce_1, cible_2) = 0.60$
$M_R (srce_5, cible_2) = 0.625$
$M_R (srce_{16}, cible_2) = 0.60$

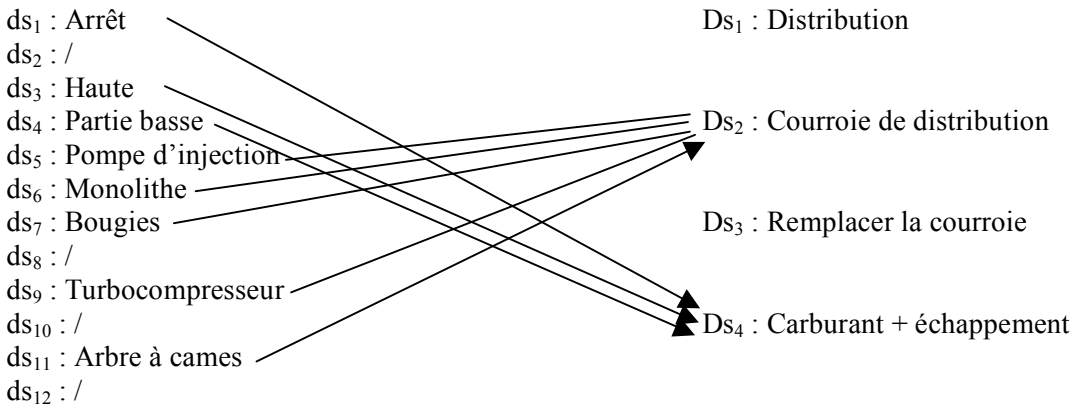
- ***Calcul de la mesure d'adaptation***

$M_A(srce_1, cible_2) = 1$
$M_A(srce_5, cible_2) = 1.43$
$M_A(srce_{16}, cible_2) = 2.33$

Par conséquent, le cas source le plus facilement adaptable au cas cible 2 est le cas source 16.

Nous pouvons remarquer que le cas source le plus similaire au cas cible 2 est le cas 5. Cependant, parmi les cas remémorés, c'est le cas source 16 qui est sélectionné pour la phase d'adaptation. Par conséquent, le cas source le plus similaire n'est pas forcément celui qu'on choisi pour l'adapter au cas cible.

Les relations de dépendance (RD) du cas source16 sont les suivantes :



D'une part, la valeur de RD du couple (Ds_2, ds_{11}) est « RD = Forte » et d'autre part, les composants « arbre à cames » et « courroie de distribution » n'appartiennent pas à la même famille. Ce qui nous amène à dire que nous sommes dans le cas de figure de « RD = Forte & différentes classes de fonctionnement ».

En appliquant l'algorithme d'adaptation, nous obtenons :

- La classe du descripteur de solution source Ds_2^{rem} est « Distribution » ;
- Le composant « arbre à cames » du descripteur cible « dc_{11} » (qui correspond au descripteur « ds_{11} » qui est en mode anormal) se trouve dans la zone commune « Carburant + échappement ». Dans cette zone, nous pouvons trouver plusieurs composants. Parmi ceux qui sont représentés dans le cas cible sont : monolithe, bougies, joint de collecteur, RGE et DVA ;
- Le composant « joint de collecteur » appartient à la même classe de Ds_2^{rem} qui est « Distribution » ;
- Substituer la valeur « arbre à cames » du descripteur « dc_{11} » par la valeur « joint de collecteur » dans « ds_{11} » qui va se répercuter sur la valeur du descripteur « Ds_2^{rem} ». De ce fait, la nouvelle valeur de Ds_2^{rem} est la suivante : Ds_2^{rem} = Joint de collecteur non serré ;
- Affecter cette valeur à « Dc_2 » : Dc_2 = Joint de collecteur non serré.

La solution sera de *resserrer le joint de collecteur qui est non serré et qui se trouve au niveau de l'intersection des deux circuits carburant et échappement*.

4.3 Troisième cas type d'adaptation : « RD = Faible »

Soit une panne qui s'est produite dans le moteur au niveau du *vilebrequin* et représentée par le cas *cible 3* (figure 8).

C3	Tourne	Mauvais			Pompe injection	Entraî-née	N				Bougies	Etin-celle	N				Turbo-compre-sseur	Prés-ence bruit	A				Courroie de distribution	Serré e	N			
10 (RD faible)	Tourne	Mauvais												Collecteur	Dég-agé	N				Vanne de reci-rculation	Pass-age d'air	N				Coussinets vilebrequin	Surf-ace lisse	N

Fig. 8. Le cas source le plus adaptable au cas cible 3.

La valeur de la mesure de similarité appliquée lors de la remémoration du cas source4 est la suivante :

- ***Calcul de la mesure de remémoration***

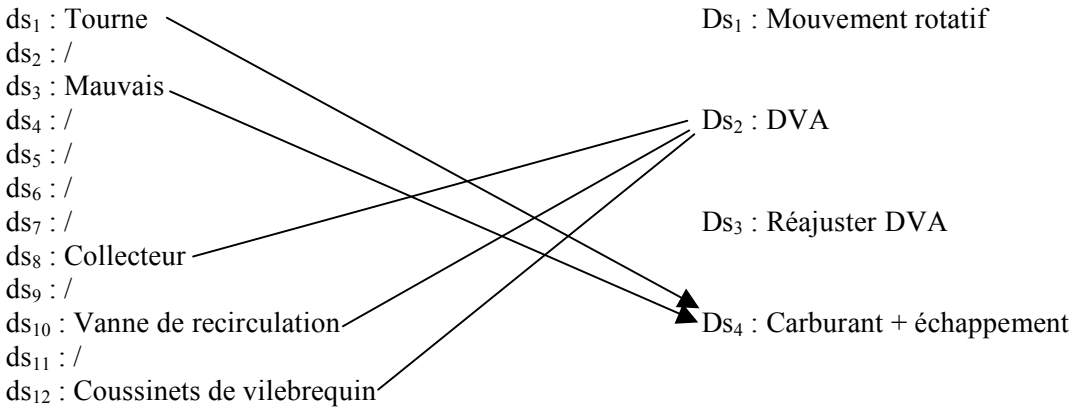
$M_R(srce_4, cible_3) = 0.75$
$M_R(srce_6, cible_3) = 0.67$
$M_R(srce_7, cible_3) = \mathbf{0.80}$
$M_R(srce_{10}, cible_3) = 0.67$
$M_R(srce_{15}, cible_3) = 0.67$

- ***Calcul de la mesure d'adaptation***

$M_A(srce_4, cible_3) = 1.2$
$M_A(srce_6, cible_3) = 1.2$
$M_A(srce_7, cible_3) = 1.33$
$M_A(srce_{10}, cible_3) = \mathbf{2}$
$M_A(srce_{15}, cible_3) = 1.6$

Par conséquent, le cas source 10 est celui qui est sélectionné pour la phase d'adaptation.
Nous pouvons également constater dans cet exemple que le cas source le plus similaire (cas source 7) n'est pas celui qui a la plus grande valeur de M_A.

La représentation des relations de dépendance entre les descripteurs de problème et les descripteurs de solution du cas source 10 est la suivante :



Nous constatons qu'il n'y a que des valeurs de RD faible. Ce qui nous amène au cas de figure suivant : « RD = Faible ».
En appliquant l'algorithme d'adaptation, nous obtenons :

- La classe du descripteur de solution source « Ds_2^{rem} » est « mouvement rotatif » ;
- Le contexte des composants du cas cible « pompe d'injection, bougies, turbocompresseur et courroie de distribution » est l'intersection des deux circuits du carburant et de l'échappement ;
- Dans cette zone, le composant appartenant au même contexte et qui appartient à la même classe de Ds_2^{rem} (Mouvement rotatif) est le composant : *vilebrequin* ;
- Substitution de la valeur « DVA » par la valeur « vilebrequin » dans le descripteur Ds_2^{rem} et mettre la valeur adéquate : Ds_2^{rem} = vilebrequin endommagé ;
- Affecter cette nouvelle valeur au descripteur Dc_2.

De ce fait, la solution donnée au cas cible serait de *changer le vilebrequin qui est endommagé se trouvant dans l'intersection des deux circuits : carburant et échappement*.

Les résultats obtenus à la suite de l'étape d'adaptation, sont probants. Ils ont été décrits sur les trois cas types existants dans un problème de diagnostic. Les autres cas de la base faisant partie de ces cas, ont donnés les résultats attendus.

5 Evaluation

Cette étude d'évaluation concernera la méthode d'adaptation mise en place. Afin de procéder à cette évaluation, nous allons réaliser des tests selon un protocole spécifique. Ce protocole concerne la division de la base de cas du moteur à explosion en deux sous-ensembles. Le premier sous ensemble contiendra 40 % des cas tirés aléatoirement de la base de cas. Ce sous-ensemble constituera la base de cas initiale (BCi). Cette dernière contiendra donc 8 cas. Les 60 % des cas restants (12 cas) constitueront le sous-ensemble cible en leur enlevant la partie solution qu'on appellera base cible (Bcib). Les cas de la base cible (Bcib) vont être soumis à la base de cas initiale (BCi) pour donner la solution adéquate.

Le but de ce protocole est de calculer la précision (accuracy) de la base de cas initiale (BCi). Afin de réaliser ce calcul, nous appliquons notre méthode d'adaptation qui est basée sur les modèles de connaissance, les mesures utilisées dans la phase de remémoration et les relations de dépendance exploitées par l'algorithme d'adaptation. Notre méthode va être comparée avec une méthode qui n'utilise pas d'adaptation. Nous indiquons que la précision (accuracy) concerne la détermination de la bonne classe de défaillance. Les résultats issus des deux méthodes sont illustrés sur la figure 9.

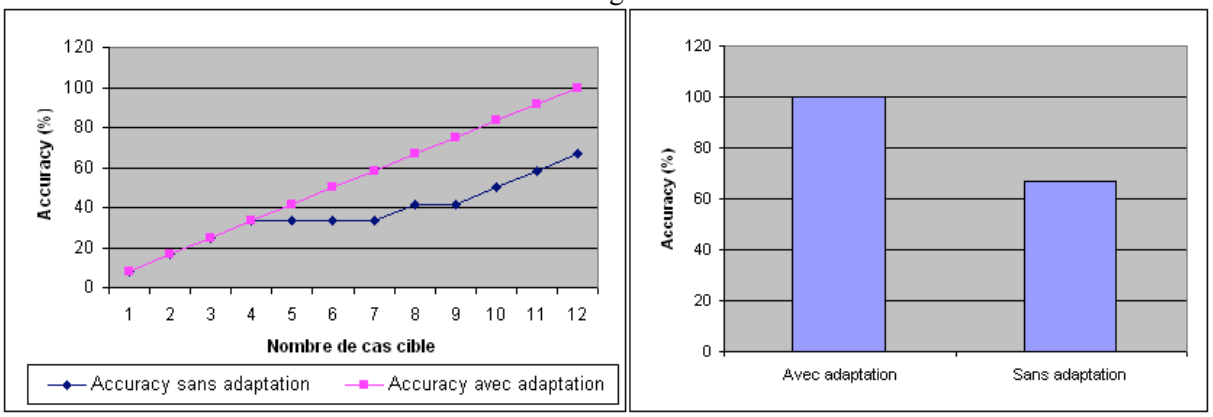


Fig. 9. Evolution de l'Accuracy suivant le nombre de cas dans la base cible (Bcib) (figure gauche), Accuracy avec notre méthode d'adaptation et sans adaptation (figure droite).

La figure 9 (droite) montre d'une part que la méthode d'adaptation mise en place a réussi une adaptation à 100% sur l'ensemble des cas cibles de la base (Bcib) présentés à la base de cas initiale (BCi). D'autre part, comme cela est montré sur la figure 9 (gauche) qui est le résultat logique de la précédente constatation, les cas cibles sont adaptés avec succès les uns après les autres avec la méthode d'adaptation mise en place.

Cependant, nous constatons que les résultats obtenus sans méthode d'adaptation sont plutôt mitigés. En effet, nous obtenons une adaptation de seulement de 66,67% sur l'ensemble des cas cibles avec une évolution non constante (figure 9 « gauche ») concernant la résolution des cas cibles.

Nous pouvons ainsi conclure que le raisonnement à partir de cas peut raisonner à partir d'un nombre restreint de cas, en prenant appui sur des modèles de connaissance et sur la phase d'adaptation ajustée à l'application considérée. De plus, la phase de remémoration, qui utilise uniquement une mesure de similarité ne tenant pas compte de l'effort d'adaptation, ne va pas sélectionner les cas les plus facilement adaptables et entre autre les cas qui ont les mêmes classes. De ce fait, sans adaptation, lorsque la classe du cas remémoré ne convient pas, cela va engendrer une mauvaise classification et donc un échec d'adaptation.

6 Conclusion

Dans le cadre d'une étude faite sur les systèmes d'aide au diagnostic industriel et de réparation, nous avons proposé un système de Raisonnement à Partir de Cas (RàPC) avec le déploiement des différentes phases. Le système conçu comprend une formalisation de cas orientée objet, associée à un modèle hiérarchique commun aux descripteurs de problème et de solution des cas de la base de cas et un modèle en relation avec le contexte d'application. Toutes les phases de RàPC dépendent de la formalisation des cas associée aux modèles de connaissance. Récemment dans [17], nous avons formalisé un cas pour un équipement de transfert de palettes SISTRE. Cette formalisation est adaptée à notre méthode. La modélisation choisie influence les mesures de remémoration et d'adaptation proposées. Cette dernière mesure est directement liée au mode de fonctionnement des composants surveillés (un attribut spécifique d'un descripteur). L'étape de remémoration développée est guidée par l'adaptation en exploitant les relations des deux mesures mises en place. Ces relations permettent de sélectionner parmi les cas remémorés ceux qui sont les plus facilement adaptables. L'étape d'adaptation exploite à son tour les relations de dépendance entre les descripteurs de problème et de solution. Ces relations de dépendance sont déterminées à travers des descripteurs pertinents ou grâce à l'utilisation du modèle de contexte reflétant les relations de cause-à-effet entre les composants de l'équipement. Nous avons ainsi détecté trois cas de figures possibles, dans l'algorithme d'adaptation, associés aux différentes valeurs des relations de dépendance.

Dans nos précédentes études [11] nous avons mis en place un système de RàPC pour un équipement de transfert de palettes. Afin de démontrer la faisabilité de notre système de diagnostic par RàPC, nous l'avons appliqué à un moteur à explosion de type 1.5 dCi K9K 105ch de la société « Renault ».

Nous envisageons dans un futur proche d'appliquer une méthode de maintenance du système de RàPC mise en place. En effet, l'évolution du système à travers le temps se fait par l'introduction de nouveaux cas suivant l'apparition de nouvelles pannes. Cette évolution peut altérer la qualité de la base de cas et que les deux mesures de remémoration et d'adaptation peuvent devenir caduques. Cette méthode de maintenance permettrait l'évolution de ces deux mesures (en insérant à titre d'exemple des poids) et de la base de cas en fonction de l'évolution du système.

7 Références

[1] A. Aamodt. Knowledge-Intensive Case-Based Reasoning and Sustained Learning. Proc. of the 9th European Conference on Artificial Intelligence, ECCBR'04, Lecture Notes in Artificial Intelligence, pp.1-15, Springer, 2004.

[2] Maintenance terminology. European standard, NF EN 13306, 2001.

[3] K. Althoff, E. Auriol, R. Bergmann, S. Breen, S. Dittrich, R. Johnston, M. Manago, R. Traphöner et S. Wess. Case-based reasoning for decision support and diagnostic problem solving: The INRECA Approach. Proc. of the 3rd German Workshop on CBR, University of Kaiserslautern, 1995.

[4] W. Cheetham et J. Graf. Case-Based Reasoning in Color Matching. Proceedings of the 2nd International Conference on Case-Based Reasoning Research and Development. Lecture Notes In Computer Science; Vol. 1266, pp. 1 – 12, 1997.

[5] B. Collins et P. Cunningham. Adaptation-Guided Retrieval in EBMT: A case based approach to machine translation. In Proceedings of EWCBR'96, LNAI 1168, 1996.

[6] P. Cunningham et B. Smyth. A Comparison of Model-Based and Incremental Case-Based Approaches to Electronic Fault Diagnosis. In: Proceedings of the Case-Based Reasoning Workshop, AAAI (American Association of Artificial Intelligence) 1994, Seattle, USA, 1994.

[7] M. Devaney et B. Cheetham. Case-Based Reasoning for Gas Turbine Diagnostics. In 18th International FLAIRS Conference (FLAIRS-05), 2005.

[8] B. Fuchs, J. Lieber, A. Mille et A. Napoli. An Algorithm for Adaptation in Case-Based Reasoning. In Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000), Berlin, Germany, p. 45–49, 2000.

[9] P.W. Grant, P.M. Harris et L.G. Moseley. Fault Diagnosis for Industrial Printers Using Case-Based Reasoning. Engineering Applications of Artificial Intelligence. Vol.9, No.2, pp.163-173, 1996.

[10] M.K. Haouchine, B. Chebel-Morello et N.Zerhouni. Adaptation-Guided Retrieval for a Diagnostic and Repair Help System Dedicated to a Pallets Transfer. In 3rd European Workshop on Case-Based

Reasoning and Context-Awareness. 9th European Conference on Case-Based Reasoning, ECCBR 2008, Trier, Germany, 2008a.

[11] M.K. Haouchine, B. Chebel-Morello et N.Zerhouni. Algorithme d'Adaptation pour le Diagnostic Technique. Actes du 16ème Atelier de Raisonnement à Partir de Cas, RàPC'2008, Nancy, France, 2008b.

[12] D.B. Leake. A. Kinley et D.C. Wilson. Case-Based Similarity Assessment: Estimating Adaptability from Experience. In Fourteenth National Conference on Artificial Intelligence, pages 674–679, Menlo Park, CA. AAAI Press, 1997.

[13] J. Lieber. 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, 1999.

[14] R. Lopez de Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M.L. Maher, M. Cox, K. Forbus, M. Keane, A. Aamodt et I. Watson. Retrieval, Reuse, Revise, and Retention in CBR. Knowledge Engineering Review, pp: 215-240, 2005.

[15] A. Mille. Raisonnement basé sur l'expérience pour coopérer à la prise de décision, un nouveau paradigme en supervision industrielle. Thèse de doctorat, Université de Saint Etienne, 1995.

[16] G. Pieri, M.R Klein et M. Milanese. MAIC: a data and knowledge-based system for supporting the maintenance of chemical plant. Production Economics, pp.1-17, 2001.

[17] I. Rasovska, B. Morello-Chebel et N. Zerhouni. Case elaboration methodology proposed for diagnostic and repair help system based on CBR, The 20th International FLAIRS Conference, Florida, 2006.

[18] M.M. Richter et S. Wess. Similarity, Uncertainty and Case-Based Reasoning in PATDEX. Automated reasoning, essays in honour of Woody Bledsoe, Kluwer, 1991, pp. 249-265.

[19] J. Rousu et R.J. Aarts. Adaptation Cost as a criterion for solution evaluation. In Proceedings of EWCBR'96, LNAI 1168, éd. Springer, 1996.

[20] B. Smyth. Case Adaptation and Reuse in Déjà Vu. Research report, 1996.

[21] B. Smyth et M.T. Keane. Retrieving adaptable cases: The role of adaptation knowledge in case retrieval. In the European Workshop on Case-Based Reasonin (EWCBR'93), pages 209–220, Kaiserslautern, Germany, November 1993.

[22] B. Smyth et M.T. Keane. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. Artificial Intelligence 102(2), 249-293, 1998.

[23] I. Watson et F. Marir. Case-Based Reasoning: A Review. The Knowledge Engineering Review, 1994.

Using case-base reasoning for database query optimization in ubiquitous environments

C. Bobineau and L. Martínez-Medina

Grenoble Informatics Laboratory
University of Grenoble, BP 72 38402 Saint-Martin d'Hères, France
e-mail: *Firstname.Lastname*@imag.fr

Abstract

Classic query optimization techniques use metadata (e. g. statistics) to estimate the cost of different possible solutions (query plans) to evaluate a query and choose the one that minimizes a cost function. This cost function determines the optimization objective, which is generally the evaluation time. In ubiquitous computing environments, where devices are autonomous and have limited physical characteristics (e.g. energy or CPU power), the evaluation time of queries is no longer the main optimization objective. Moreover metadata required for a priori estimating the evaluation cost of query plans are not always available. In order to solve this problem, this article describes a new query optimization technique based on automatic learning, particularly on case-based reasoning. This technique is not another cost function-based query optimization technique. It considers a query case base where cases represent experiences of queries that have been previously executed, optimized and evaluated. It address the exploitation of the case base for generating execution plans and learning on existing ones according to measures inserted in the case base. Currently we are working in the prototype that implements the query optimization techique that we propose.

1 Introduction

An ubiquitous computing environment is created from the integration of mobile or stationary computational resources (software applications, web services, computation tools, database management systems, etc.) and electronic devices (sensors, PDAs, cell phones, etc.) [3]. These resources and devices offer mechanisms to store, process and share heterogeneous information in a distributed fashion [11]. An ubiquitous computing environment must provide a set of mechanisms that allow the user to access any information at any time, from anywhere and from any computing device. Query optimization is essential to improve the performance of these methods. The resources composing an ubiquitous environment are [5]:

1. *Heterogeneous*. Ubiquitous computing environments integrate a variety of computational resources and electronic devices, like portable computers, PDAs, cell phones, censor networks, embedded systems, among others. These devices present important physical and logical differences.

2. *Dynamic*. Resources in ubiquitous environments change continuously, many of these changes are due to their mobility, e.g. when a resource is moved, its location and availability change. Also, the communication network properties and the set of resources with which it interacts are different.

3. *Distributed*. Resources in ubiquitous environments are distributed in different locations within a physical environment to which the user interact to carry out a set of activities.

4. *Autonomous*. Every resource integrated in the environment is autonomous; this means that it can change its availability status at any time (it can be available to interact with other resources within the environment or not) .

5. *Physically constrained*. Resources present physical limitations that constrain their appropriate operation, e.g. processing and storage capability, energy consumption, location, among others.

6. *Lacking of metadata*. Distributed systems related to data management and data retrieval are based in the notion of global schema. It is a unified and integrated representation of the local schemas exported by each source to represent its own data. In ubiquitous computational environments maintaining such a global schema is very expensive due to the constant changes in the environment. Additionally, the continuing changes presented by the resources do not allow to associate static metadata to data sources.

Ubiquitous computing environments must provide a set of methods to retrieve information from available resources. The properties that characterize resources in ubiquitous environment represent new challenges for query processing. Some of them are related to the variable execution context and physical constrains as result of their dynamicity and autonomy. Others refer to the absence of metadata used to estimate the cost of execution plans (possible execution plans to compute the results of the query). The need to optimize a query according to different parameters (user requirements) and not just according to execution time represents another significant challenge.

In this work we propose an optimization strategy that is capable to address the necessity to optimize a query in a context in which metadata acquisition and maintenance is very costly and difficult. Also we address the necessity to optimize a query according to different parameters of interest. Our work suggests a query optimization technique that does not require metadata and that is capable to estimate the cost of an execution plan according to the user requirements.

2 Impact of ubiquitous computing on query optimization

Query optimization is essential to improve query processing efficiency. Classical query optimization techniques are not capable to deal with the challenges that resource characteristics in ubiquitous environments imply. They typically generate query execution plans taking in to account the optimization of a single dimension, query execution time [6]. In ubiquitous computing environments, where devices are autonomous and have limited physical characteristics (e.g. energy), the evaluation time of queries is no longer the main objective of optimization. Furthermore, classical query optimization techiques based the execution plan evaluation on metadata that is not available [13] [4]. We propose an approach for query optimization in ubiquitous computing environments based on machine learning techniques in order to address the problem related to lacking of metadata [5].

2.1 Classical query optimization techniques

In order to understand the challenges that implies query optimization in ubiquitous computing environments, particularly those challenges that correspond to metadata absence; we present the general notion of classical query optimization technique. Basically, it consists in three main steps, generate all possible execution plans that solve a query, estimate the cost of each plan (by applying a cost function) and finally, select that one that minimizes the optimization objective [6]. We provide a typical declarative query Q over a data set accrording to the database schema *DBS*.

DBS:
Restaurant (numRest, nomRest, specialite, adresse, ville)
Ville (nomVille, superficie, region)
Region(nomRegion, departement)

Q:
Select Restaurant.nomRes, Restaurant.adresse, Restaurant.ville
From Restaurant, Ville, Region
Where Ville.region = ŞRAŤ and Restaurant.specialite = ŞItŤ and
Restaurant.ville = Ville.nomVille and Ville.region = Region.nomRegion

Table 1 presents the attributes that are involved in operations of Q. Those attributes are *region* (a_1) and *ville* (a_3) that pertain to *Ville* relation, *specialite* (a_2) and *nomVille* (a_4) that pertain to *Restaurant* relation and *nomRegion* (a_5) that pertains to *Restaurant* relation.

Id	Name	Table
a1	region	Ville
a2	specialite	Restaurant
a3	ville	Restaurant
a4	nomVille	Ville
a5	nomRegion	Region

Table 1: Where operations attributes

Figure 1 presents the classical query evaluation process that must be performed in order to evaluate the query Q. This process is composed by four modules: query parser, query optimizer, code generator and query executor. We are interested in analyze the optimizer characteristics since the optimization of a query is carried out by this module.

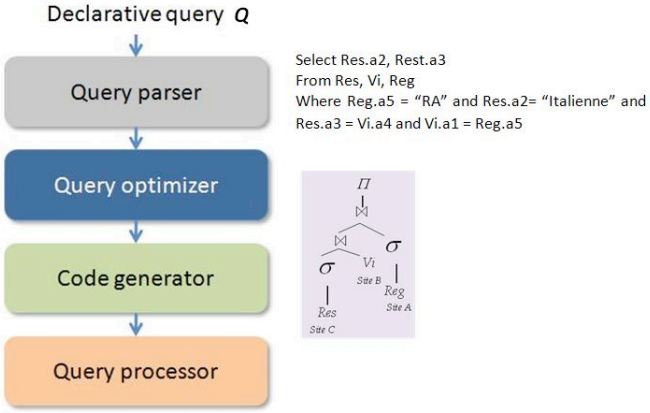


Figure 1: Query evaluation process[6]

2.2 Query optimizer

Figure 2 presents the internal query optimizer components. It is composed by tree sub-modules: logical optimizer, global optimizer and physical optimizer. Logical and physical optimization are related to centralized environments. Global optimization is required in distributed execution environments. The explanation of each module within query optimizer aims to make clear that each classical optimization step requires metadata to be accomplished.

1. *Logical optimizer*. Logical query optimization aims to reduce the number of data items combined as intermediate results. The lower number of data items are manipulated, the lower the resources consumption will be to load/store and compute them. It is necessary to decide the best order to apply for example selection, projection and join operators, metadata, such as data cardinalities, are required for this computation.

2. *Global optimizer*. Global query optimization aims to minimize communication cost by deciding where computations will be done and thus which data must be transferred from one device to one other. This process involves metadata, such as communication cost related to interactions among resources.

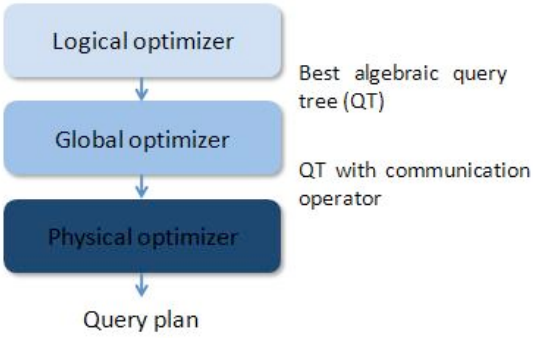


Figure 2: Internal modules of a query optimizer

3. *Physical optimizer*. Physical optimization techniques aim to minimize local evaluation cost (generally disk access and thus execution time) to perform algorithms related to each operator that is included in the query execution plan. To estimate the execution cost of each algorithm and to minimize the disk access, metadata related to execution context are required.

3 Learning-based query optimization

The general idea of this approach is to learn from experience obtained as a result of previously executed queries. We base our approach on machine learning techniques [10]. Machine learning is an artificial intelligence subdiscipline concerned by the design and development of methods that allow computers to learn automatically in order to improve or create specific capabilities. Typically, these methods are based in information obtained as a result of experiences related to problem solving and execution environment behavior. There are different automatic learning methods, we are interested in case-base reasoning [9].

According to the query optimization process that we propose, given a query Q an existen query plan must be retrieved if it can be adapted to the new query and if there exist the resources (e.g. memory, CPU and enery) to acomplish its execution. If this is not the situation, a new execution plan must be generated, we provide more details related to this processes in sections 3.2 and 5.2 respectively.

3.1 Basic concepts of case-base reasoning

Case-base reasoning is a machine learning approach that consists of a process to solve new problems based on the experience of similar problems already solved (cases). A case is the unit of knowledge and consist of a problem description, its solution, and, typically, annotations about how the solution was derived. Cases are stored in a case base. Case-base reasoning has been formalized for purposes of computer reasoning as a four-step process: retrieve, reuse, review and retain [2].

In retrieval step, relevant cases to solve a new problem must be retrieved. Then, in reusing step, a mapping between the solution offered by the relevant case and the problem requirements must be done, this may involve adapting the solution as needed to fit the new situation. Having mapped the previous solution to the target situation, the new solution must be tested in the real world (or a simulation), this is carried out in the reviewing step classically involving a human expert. Finally, after the solution has been successfully adapted to the target problem, the new experience must be stored as a new case in memory; this is accomplished in the retaining step.

3.2 Case-based reasoning adaptation to query optimization technique

We propose query optimization strategy that adapts case-base reasoning in order to provide better and better execution plans to solve new queries. This strategy recovers, adapts or generates execution plans using the

knowledge acquired from previous experiences to optimize and execute similar queries [10]. According to this approach, a case corresponds to the representation of the knowledge acquired from the experiences obtained from the optimization and evaluation of previous queries. A problem corresponds to a new query submitted in the ubiquitous computing environment and a specific optimization objective. The solution is represented by the corresponding execution plan.

3.2.1 Query representation

A query associated to a problem is defined by three clauses *selectClause*, *fromClause* and *whereClause*. The *selectClause* specify the attributes that must be projected as query result. The *fromClause* specify the data sets that contain the information requested in the query. The *whereClause* specify the set of conditions (for data selection and data combination or join) that must be verified by the data to form part of the query result. Figure 3 illustrates the query representation. In a query, selection and join operations are the most important and most frequent.

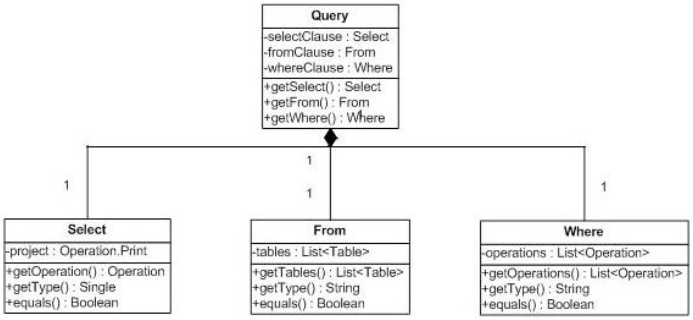


Figure 3: Query representation (UML diagram)

Each operation is defined by a type e.g. Select or Join, a set of attributes and a specific condition. Selection and join are the operations which are relevant for physical query optimization. Selection operation is expressed by means of a condition of the form *condition(attrexp, cnstexp)*, where *attrexp* represents the selection attribute and *cnstexp* a suitable constant value. Join operation is expresed by means of a condition of the form *condition(attrexp.a, attrexp.b)*, where *attrexp.a* and *attrexp.b* represent the join attributes. The condition express a comparison (i.e. equal, different, lower, greater, etc.) between the values of a table data correspondig to the *attrexp* attribute and the specified constant value or to another attribute. An attribute is defined by an id, its name and the table to which it pertains.

We can use the query Q defined before, to illustrate the *whereClause* representation. Table 2 presents the operations involved by Q. There are two selection operations o_1 and o_2, as well as two join operation o_3 and o_4. They involve the attributes presented in Table 1. They apply the the condition Equal that express the comparison operator of equality. The operation set related to Q is denoted by $Q_s = \{o_1, o_2, o_3\}$. It is possible to apply different conditions to the same attribute(s), i.e. o_x = *Greater(R.*a_n*,value)* and o_y = *GreaterOrEqual(R.*a_n*, value)*, select the values that are greater/greater or equal to some specific attribute an that pertains to the relation *R*. We propose the concept of operation family in order to grouping the operations with these characteristics.

Id	Description	Type	Condition
o1	`Ville.region = 'Rhone Alpes'`	`selection`	`Equal(a1,Rhone Alpes)`
o2	`Restaurant.specialite = 'Italienne'`	`selection`	`Equal(a2,Italienne)`
o3	`Restaurant.ville = 'Ville.nomVille'`	`join`	`Equal(a3,a4)`
o4	`Ville.region = 'Region.nomRegion'`	`join`	`Equal(a4,a5)`

Table 2: Where operations attributes

We propose the concept of operation family in order to group operations with these characteristics. Two operations o_x and o_y pertain to the same operation family if they are of the same type (selection or join) and involve the same attributes (each of them must pertains to the same data source respectively). An operation family is represented as follows:

$$\Im_{R.an} = \{o_n \mid o_n = condition(R.a_n, value)\} \quad (1)$$

The operation family $\Im_{R.an}$ is composed by the operations set o_n of the form *condition(R.a_n, value)*, where a_n is an attribute that pertains to the relation *R* and the condition denotes the set of all possible comparison operators: Equal, EqualOrLower, Lower, GreaterOrEqual, Greater and Different. All the queries are asociated to a set of operation families. The *whereClause* of a query *Q* is defined by a set of operation (sort, select and join). This operations are members of different operation families: $\Im_{R1.a1}$, $\Im_{R2.a2}$ and $\Im_{R1.a3,R2.a4}$. Equation (2) shows the operation families Q$\Im$ that are associated to each operations in Q.

$$Q\Im = \{\Im_{R1.a1}, \Im_{R2.a2}, \Im_{R1.a3,R2.a4}\} \text{ and } \Im_{R2.a4,R3.a5}\} \quad (2)$$

Queries must be classified according to the operation families to wich pertain the operations that the query includes. Each different combination of $\Im_{R.an}$ conforms a class description, i.e. the class C_n that is defined by the following operation families in (3). The class C_n is composed by all queries Q_n that contains at least one operation that pertains to each of the specified families as is defined in (4). This means, a query Q_n pertains to the class C_n if and only if for all operation family $\Im$ that describes C_n, exists an operation on in Q_n such as this operation is of the form of the operation family $\Im$.

$$C_n = \{\Im_{Rn.an}, \Im_{Rm.am}, \Im_{Rn.ap,Rm.aq}\} \text{ and } \Im_{R2.a4,R3.a5}\} \quad (3)$$

$$Q_n \in C_n \text{ iff } (\forall \Im_{Rn.an} \in C_n)\ \exists\ ((o_n \in Q_n) \wedge (o_n \in \Im_{Rn.an})) \quad (4)$$

In order to clarify the previous equations we present an illustrative example. Refering the query example provided in section 3 the selection operation o_1 pertains to operation family $\Im_{Ville.region}$, the selection operation o_2 pertains to operation family $\Im_{Restaurant.specialite}$ and the join operation o_3 pertains to opreation faimly $\Im_{Restaurante.vile,Ville.nomVille}$. The operator and the attribute value are not important to determine de operation faimily to which an specific operation pertains, the important knowledge is related to the operation type and the attribute(s) included in the operation. The operation families described before make up a class (a). Any query that is composed by operations that pertains to the families described before pertains to the same class (b).

$$C = \{\Im_{Ville.region}, \Im_{Restaurant.specialite}, \Im_{Restaurante.ville,Ville.nomVille} \text{ and } \Im_{Ville.region,Region.nomRegion} \quad (a)$$

$$q \in C \text{ iff } (\forall \Im_{Rn.an} \in C_n)\ \exists\ ((o_n \in q) \wedge (o_n \in \Im_{Rn.an})) \quad (b)$$

A query is the medullar part of knowledge in the definition of a problem and a case. The problem specifies a query that must be optimized, the optimization parameters (e.g. memory, energy and CPU) and measures related to the computational resources that are available for query execution. The case specifies the query that was optimized, the solution to solve this query (query plan) and the measures related to the computational resources that were consumed by the query execution. It is clear that the problem is not only the query by itself because it also includes the circumstances under the query must be solved and the optimization objective, it varies according to user requirements.

The query is the piece of knowledge that links a problem with the existent cases, is to say, the cases that contain a query that is similar to the query included by the problem can be useful to solve the new query, but this is not all that is important for us. Also is necessary to put attention on the computational resources consumed by the query and those that are available at the moment that the new query will be executed as well as in the optimization objective that can change each time that the query is executed.

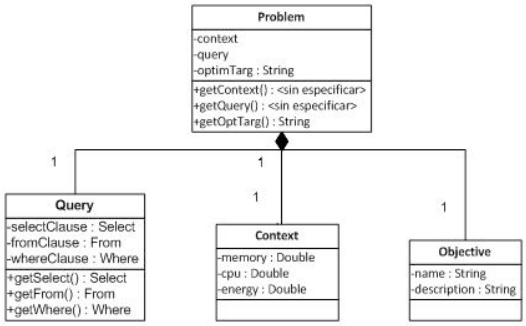


Figure 4: Problem representation (UML diagram)

3.2.2 Problem representation

A problem is composed of a query, a context execution representation and an optimization objective. Figure 4 illustrates the components of a problem.

In this representation, the query is the optimization target that must be solved. The context is described as a set of couples of the form *<attribute, value>* that represents the characteristics of the execution environment at the moment of the query evaluation. These attributes may include CPU charge, available memory, and remaining energy, among others. Finally, the optimization objective indicates the resource or resources that want to be optimized, e.g. minimize energy consumption. The optimization objective is denoted by the cost function and reflected by the execution plan when it is adapted since the relevant case is retrieved taking into account the resources that must to be optimized according to the user or application requirements.

3.2.3 Case representation

A case is composed of a query, a solution (query plan) and a set of evaluation measures that are used to express the optimization objective of a query. Figure 5 illustrates the components of a case.

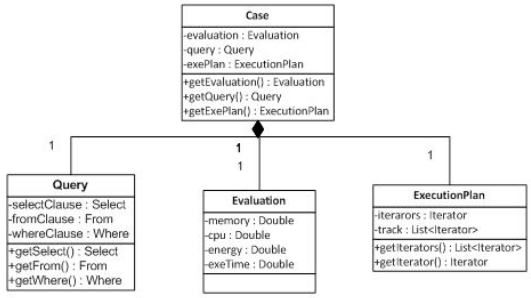


Figure 5: Case representation (UML diagram)

In a case, the query corresponds to an optimization target that has been evaluated and solved. The solution corresponds to the physical execution plan that solves the query, typically, a tree where nodes represent algorithms (implementations of query operators) and links represent data flows among operators. Finally, the evaluation corresponds to a set of measures collected during the query execution. These measures are represented as couples of the form *<attribute, value>* and express the computational resources(e.g. memory, CPU or energy) consumed by the query execution.

3.2.4 Reasoning process

The reasoning process that must be accomplished given a new problem is an adaptation of the four-step case-based reasoning process. Recovering step is based on a similarity function in order to perform a smart search

to retrieve the most relevant cases to solve the query of the problem. Among these relevant cases, the one that minimizes the cost function of the problem is selected. Reusing step is related to the adaptation process of the execution plan included in the relevant case to the new query. Reviewing step consist in verifying the query by its execution during which resource consumption measures are done. Finally, in the retaining step, the problem and its solution are stored in the case base in form of a new case [7].

1. *Retrieval*. Relevant cases are determined by a similarity function applied directly over the queries, in order to know which queries are similar and useful to solve the new problem (query, optimization objective and context measures). Given a problem, those cases that contain a similar already solved query must be retrieved. If there is no relevant case in the case base, a new query plan must be (pseudo-)randomly generated to be able to increase the query optimizer knowledge.

2. *Reuse*. It is possible to reuse the retrieved solutions by means of their adaptation to the new situation. This adaptation process concist in execute the pertinent changes to the query plan included by the retrieved case (relevant) in order to satisfy the query problem requirements, e.g. modifying the projection attributes or the comparison values.

3. *Review*. In this step the execution plan is verified. We analyze and adopt a specific model, the iterator model, in order to base the execution of each operation involved in a query according of the proposed algorithms. Iterators are responsible of applying specific algorithms implementing query operators and of taking some resource consumption measures.

4. *Retention*. The query class, query plan and consumption measures are stored in form of a case within the case base.

4 Similarity function

We propose a similarity function that is performed in two steps. The class to which a query pertains is determined in the first step; this is achieved by an inter-class similarity function. Then, the most relevant case within the class must be retrieved by means of an intra-class similarity function. It is based on the contrast model of similarity proposed by Tversky [14] that allows determining the similarity between two objects by means of a feature-matching function. Similarity increases as most common features and decreases as most distinctive features [1]. The formalization of the original definition is expressed as follows [14]:

$$S\,(a, b) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A) \quad (5)$$

Similarity between a and b, is defined in terms of the features common to a and b, $A \cap B$, the features that pertain to a but no to b, A - B, and those that pertain to b but no to a, B - A. The function f measures the salience of a particular set of features (also can be a single feature)[14].

In our proposition, similarity is defined as an increasing function of common operation families and as a decreasing function of different operation families, that is, families that pertain to one query but not the other. The formalization of this definition in terms of the similarity between a query and a class is expressed as follows.

$$S\,(C_1, Q) = \theta\, \Im(C_1 \cap Q) - \alpha\, \Im(C_1 - Q) - \beta\, \Im(Q - C_1) \quad (6)$$

Similarity between C_1 and Q, is defined in terms of operation families common to C_1 and Q, $C_1 \cap Q$, the features that pertain to C_1 but no to Q, C_1 - Q, and those that pertain to Q but no to C_1, Q - C_1. The function f refers particularly to operation families $\Im$, which in this case, are the features that must be compared.

Intersection result between query Q and each class C_1, C_2 and C_3 illustrates the common operation faimilies between C_q and the other classes. From these results, it is possible to state that Q pertains to the class C_1. The class C_2 is less similar to Q than C_1. Finally, the less similar class with respect to Q is the class C_3. We can conclude that the query class of Q already exist in the case base and is C_1.

Intra-class similarity function aims to find the most similar queries with respect to a new query, which is desired to be optimized, within the same class. Similarity between two queries Q_1 and Q_2 is defined as an increasing function of common operations (identical operations in terms of its type, attributes and operators). The formalization of this definition is as follows:

$$S(Q_1, Q_2) = \theta o(Q_1 \cap Q_2) - \alpha o(Q_1 - Q_2) - \beta o(Q_1 - Q_2) \quad (7)$$

Similarity between Q_1 and Q_2 is defined in terms of the operations that are common to Q_1 and the features that pertain to Q_1 but no to Q_2. It is possible to establish a mapping one to one of each of the operations included in Q_1 and Q_3 according to the comparison operator that each of them apply. Q_1 and Q_2 have two operations in common, they differ in the operator applied by the join operation. According to this simple analysis, Q_3 is the most similar query with respect to Q_1 because contains the maximum number of operation mappings.

5 General architecture of a query optimizer based on case-base reasoning

We propose a query optimizer that provides two alternatives in order to solve a query. The fist one reutilize the solutions related to queries that have been solved, the second one generate new solutions if there is no relevant known solution. The optimizer is composed of two main modules, the case-reasoner and the execution plan generator. The case-base reasoner is in charge of adapting the solutions of similar queries to the new situation. The execution plan generator is in charge of generating new execution plans in a pseudo-randomly fashion. The case-base reasoner is the most complex of the two modules but the smartest, on the other hand, the execution plan generator is more simple and probably faster but it does not apply machine learning techniques. Figure 6 illustrates the optimizer architecture.

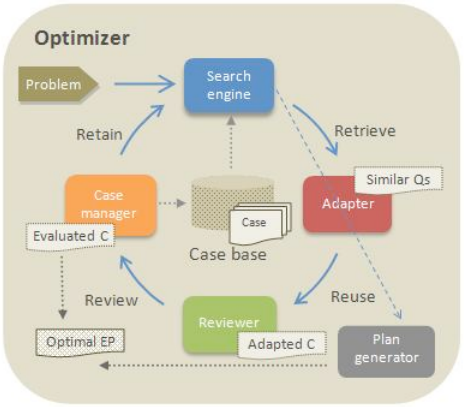


Figure 6: Optimizer architecture

5.1 Case-based reasoner

The case-based reasoner implements the four case-base reasoning steps. It is composed by four submodules, one module for each reasoning step. The modules that comprise it are the smart search engine, the adapter, the execution engine and the case-base manager. The last one interacts with a case base that allows to manage the useful knowledge acquired from experiences(queries that have been solved).

5.1.1 Smart search engine

It is in charge to accomplish the fist step of reasoning proccess, retrieve relevant cases. This module performs a smart search by applying the similarity function (inter-class similarity and intra-class similarity) in order to

recover those cases that contains the most similar query with respect to the query target to be optimized. Additionally, this module selects among those retrieved cases that one that minimizes the optimization parameters (e.g energy consumption).

5.1.2 Adapter

It is in charge of adapting the query plan included in the relevant case (retrieved in the anterior step) to the query problem specifications. The relevant case is the most convenient (similar case) to facilitate and minimize the cost of the adaptation process, however, there exist different similarity levels between the query related to the relevant case and the query problem.

When a relevant case is retrieved, a detailed comparison between the clauses of the new query and the relevant query (the query included by the relevant case) is carried out. This determines a similarity level between the two queries. This level indicates which clauses of the relevant query must be adapted. This adaptation can be performed just over select and where clauses. In the case of the selection clause, the interesting attributes to be projected can vary. In the case of the where clause, the comparison operators or some values related to the variables can be modified. On the other hand, the from clause can not be changed because the tables to be queried can not be change. Table 3 illustrate them.

Sim. Level	Equivalent clauses	Different clauses
4	`selectClause, fromClause and whereClause`	`---`
3	`fromClause and whereClause`	`selectClause`
2	`selectClause and fromClause`	`whereClause`
1	`fromClause`	`selectClause and whereClause`
0	`---`	`fromClause, selectClause and whereClause`

Table 3: Similarity levels between two queries

The adaptation must be performed for the similarity levels (3), (2) and (1). If the similarity level is (3) the from and where query clauses are equal, the adaptation must be performed on the select clause, this means that the attributes to project must be actualized according to the new requierements. If the similarity level is (2) the from and select query clauses are equal, the adaptaton must be performed on the where clause, this means that some operation conditions are different but are applyed over the same attributes and the same tables, in this case, the condition type (e.g. equal, different and greater) and/or the constant value to which the attribute is compared must be adjusted. Finaly, if the similarity level is (1) an adaptation of the select and where clauses must be made. When the level is equal to (4) or (0) it is not needed to perform an adaptation, in the first case because the execution plan is useful to solve the new query, in the second case because it is no possible to carry out an adaptation according to our approach. As a conclusion, this similarity level refect the necessary adaptations that must carry out since it determines the query clauses that must be adapted.

As a conclusion, the similarity level reflect the necessary adaptations or adjustments that must be carried out and consecuently, it allows us to determine how easy is to adapt an existent query solution to the new query specifications. This represent a direct relation between a problem and the solutions.

5.1.3 Execution engine

It is in charge to execute the new query execution plan that is provided by the adaptation module. In this step, some measures (e.g. memory, cpu and consumed energy) related to the resource consumption of the new query plan are done.

5.1.4 Case base manager

It allows retaining the new knowledge in form of a new case or adjusting existing ones. In order to store a new case, it is necessary to determine if it pertains to some existing query class or if a new one must be constructed. The similarity function is useful also in this step because it determines the similarity between two classes.

5.2 Execution plan generator

It is possible that there is not any (or not enough) relevant case to solve a specific problem; a totally new query execution plan must be built. We propose a pseudo-random mechanism for the generation of execution plans that consist of a single phase. In contrast to typical optimization strategies, this mechanism does not include the construction of algebraic trees; it constructs directly trees composed by physical operators. We implement these operators according to the iterator model. An execution plan is reasonable if it does not include redundant and useless operations that represent a waste of computational resources.

The construction of reasonable execution plans are based in a set of rules that define the physical operator (algorithm based in the iterator model) that is pertinent to apply for the execution of each operation. The set of rules implements classical algebraic heuristics (selections and projections first) and identify reasonable choices that have to be randomly done (e.g. join execution order or reasonable algorithms). These rules are applied in each level of the execution plan, is to say, each time that an operator is added to the tree.

6 Related work

Some existing works use machine learning techniques in order to address the query optimization problem. However, they are still based on metadata. More over, they perform some special treatment over metadata in order to improve the optimization calculus, i.e. the LEarning Optimizer LEO that allows to repair incorrect statistics and cardinality estimates of a query execution plan[12].

Optimizers based in plan caching i.e. SQL Server also use machine learning techniques. Particularly, SQL Server stores execution plans and data buffers. Nevertheless, it extends the general query optimizer architecture, in addition to this, it is based on classical query optimization techniques, that is to say, it is stil tightly tied up to the use of metadata[8].

7 Conclusions and future work

We propose a new query optimization technique that exploits case-based reasoning in order to improve query optimization in ubiquitous environments. Our approach deals with the challenge that the lack of metadata implies in this execution context. We propose a technique that is based in the useful knowledge (resource consumption measures) obtained from previous query executions. In addition, we propose a technique that allows the configuration of the optimization objective according to the users and application requirements, even for each single query.

The most important contributions of our work are centered in the reasoning steps related to retrieval and re-adaptation of the useful knowledge. These steps retrieve the most relevant cases and adapt a previous solution to the new situation. We propose a model to represent a query, a problem and a case. Moreover we define a similarity function to determine which query related to cases stored in the case base are similar and useful to optimize a new query, which is the problem to be solved. The adaptation depends on the similarity level that is determined between the two queries. Basically, the adaptation process consist in evaluating the operations of the query with variations in their conditions. Finally, we propose a pseudo-random query plan generator to be able to evaluate queries even if there is no corresponding case in the case base.

Currently we are working in a prototype that implements our solution in order to perform an experimental evaluation of our approach. It is very interesting to apply this solution to different application domains where possible solutions can be enumerated, performance criteria are known but where important informations permitting to decide what is the best solution are not available (statistics in our case). For example, automatic service

composition generated on demand via a declarative language seems a good target. Dynamicity management is also an important point to work on. The system should be able to detect that cases in its case base are no more relevant and thus delete them. This is a problem of knowledge refreshment. We are also working on improving the knowledge acquisition and exploitation, for example by sharing case bases between nodes in a network or by reducing the granularity of cases to handle subqueries instead of full queries.

References

[1] Features of similarity and category-based induction, 1997.

[2] A. Aamodt et E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[3] E. Bardram. Activity-based computing: support for mobility and collaboration in ubiquitous computing. *Personal Ubiquitous Comput.*, 9(5):312–322, 2005.

[4] G. Abowdand A. Deyand, P. Brown, et et al. Towards a better understanding of context and context-awareness. In *In: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC 1999)*, September 1999.

[5] M. Franklin. Challenges in ubiquitous data management. In R. Wilhelm, editor, *Informatics - 10 Years Back. 10 Years Ahead.*, pages 24–33. Springer-Verlag, 2001.

[6] Y. Ioannidis. Query optimization. *ACM Comput. Surv.*, 28(1):121–123, 1996.

[7] L. De Mantaras, R. McSherry, et et al. Retrieval, reuse, revision and retention in case-based reasoning. *Knowl. Eng. Rev.*, 20(3):215–240, 2005.

[8] Microsoft. Excution plan caching and reuse. http://technet.microsoft.com/en-us/library/bb545450.aspx, 2008.

[9] J. Nilsson. *Introduction to Machine Learning (An Early Draft to a proposed textbook)*. 1996.

[10] M. Othman-Abdallah. *Optimisation de requêtes par apprentissage*. PhD thesis, Institut National Polytechnique de Grenoble.

[11] F. Perich, S. Avancha, et et al. Profile driven data management for pervasive environments. In *In: Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA 2002)*, pages 361–370, London, UK, 2002. Springer-Verlag.

[12] M. Stillger, G. Lohman, V. Markl, et M. Kandil. Leo - db2's learning optimizer. In *In: Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 19–28, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[13] D. Subramanian et K. Subramanian. Query optimization in multidatabase systems. *Distrib. Parallel Databases*, 6(2):183–210, 1998.

[14] A. Tversky et I. Gati. Studies of similarity. 1978.

Utilisation d'une algèbre temporelle pour la représentation et l'adaptation de recettes de cuisine

Florence Le Ber[1,2], Jean Lieber[2], Amedeo Napoli[2]

[1]LHyGeS UMR 7517 – ENGEES, F-67000 Strasbourg
florence.leber@engees.u-strasbg.fr
[2]LORIA UMR 7503, F-54500 Vandœuvre-lès-Nancy
{Jean.Lieber,Amedeo.Napoli}@loria.fr

Résumé

Nous traitons ici de la représentation du temps et du raisonnement temporel qualitatif dans la cadre du raisonnement à partir de cas. Nous nous appuyons pour cela sur l'exemple jeu d'une application culinaire, ayant pour but de générer des recettes opérationnelles. Les suites d'opérations sont représentées par des graphes où les arêtes sont des relations entre intervalles temporels. L'adaptation des recettes peut alors s'effectuer grâce à des techniques de résolution de contraintes. Nous décrivons quelques exemples où ces techniques peuvent être mises en jeu. Nous concluons en situant l'approche proposée par rapport aux travaux combinant RàPC et représentation du temps ou planification.

Mots clés : modèles qualitatifs du temps, intervalles de temps, durée, résolution de contraintes temporelles, planification, TAAABLE.

1 Introduction

Cet article est une réflexion sur l'utilisation de modèles qualitatifs du temps dans le raisonnement à partir de cas. Nous prenons pour exemple d'application le projet TAAABLE [5], sur la représentation, la remémoration et l'adaptation de recettes de cuisine. L'objectif de cette application est de générer des recettes opérationnelles, à partir d'une liste d'ingrédients et de contraintes, en s'appuyant sur une base de recettes données. L'article est organisé comme suit. La section 2 introduit des modèles qualitatifs du temps. La section 3 propose une représentation de recettes de cuisine sous forme de graphes d'actions, tandis que la section 4 aborde les problématiques de remémoration et d'adaptation des recettes ainsi représentées à travers différents exemples. La dernière section est une discussion suivie d'une brève conclusion.

2 Quelques éléments sur la représentation du temps

La modélisation du temps est une thématique déjà ancienne en intelligence artificielle. Elle s'applique à la représentation de connaissances, la compréhension de la langue naturelle, le raisonnement de sens commun, le raisonnement qualitatif, le diagnostic ou la planification. Différents modèles ont été développés et peuvent se décliner selon différents aspects : temps ponctuel ou sous forme d'intervalle, temps totalement ou partiellement ordonné (temps linéaire ou branché), temps discret ou dense, borné ou non, incluant différentes granularités ou non [14]. Nous nous focalisons ici sur les approches qualitatives.

2.1 Algèbres temporelles

Les principales représentations qualitatives du temps sont l'algèbre d'intervalles de Allen [1, 2] et l'algèbre de points de Vilain et Kautz [21]. Dans la théorie d'Allen, 13 relations de base (ou relations atomiques) décrivent toutes les manières possibles d'ordonner les extrémités de deux intervalles (figure 1). Par exemple, pour $X =$

$[X^-, X^+]$ et $Y = [Y^-, Y^+]$ deux intervalles non réduits à un point, $X^+ < Y^-$ s'écrit $X\ \{b\}\ Y$ (pour *before*), $X^+ = Y^-$ s'écrit $X\ \{m\}\ Y$ (pour *meet*) et $X^- < Y^- < X^+ < Y^+$ s'écrit $X\ \{o\}\ Y$ (pour *overlap*). Les relations de base peuvent être combinées par disjonction, on obtient alors 2^{13} relations. Il n'y a pas de représentation de la durée des intervalles. Dans l'algèbre de points de Vilain et Kautz, trois relations de base, précède ($<$), identique ($=$) et suit ($>$), sont considérées.

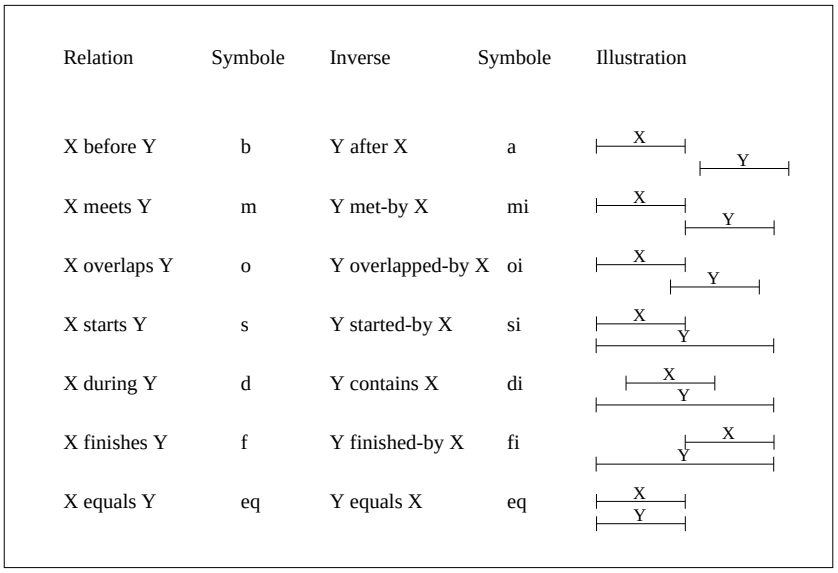


FIG. 1 – Les treize relations de Allen entre deux intervalles de temps X et Y.

Les tâches de raisonnement (vérifier la consistence, trouver une interprétation, inférer de nouvelles relations) sont NP difficiles en algèbre des intervalles et polynomiales en algèbre des points. Les relations entre intervalles peuvent être pour parties traduites en relations entre points, ce qui permet de résoudre les tâches exprimées dans une algèbre d'intervalles réduite. L'inférence des relations s'appuie sur les règles de composition, telle que $X\ \{b\}\ Y \wedge Z\ \{d\}\ Y \rightarrow X\ \{b\}\ Z$ pour trois intervalles quelconques X, Y, Z (si X est avant Y et Z est pendant Y alors X est avant Z). À partir de la table de composition des relations atomiques, on peut déduire le résultat de la composition de deux relations quelconques [2]. Au-delà, les techniques de résolution de contraintes peuvent s'appliquer pour vérifier la cohérence d'un ensemble de relations entre n intervalles et trouver un scénario [7].

2.2 Prise en compte de la durée

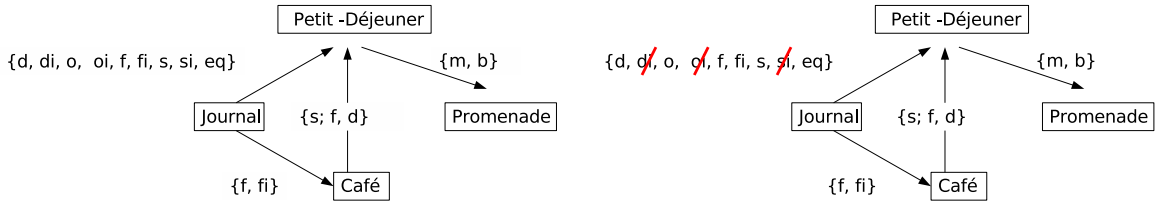
Le modèle $INDU$ pour *interval and duration* [17] introduit une notion de durée des intervalles. Ce modèle s'appuie sur les treize relations de Allen et les complète par trois relations permettant de comparer la durée de deux intervalles X et Y : X est de durée inférieure à Y ($X\ \{<\}\ Y$), supérieure ($X\ \{>\}\ Y$) ou égale ($X\ \{=\}\ Y$). Les relations composées s'écrivent sous la forme suivante : $m^<$, $m^>$, $m^=$ par exemple pour la relation *meets*. On obtient ainsi 25 relations car certaines combinaisons sont impossibles ; par exemple la relation eq ne se combine[1] qu'avec la relation $=$.

D'autres modèles permettent de représenter des intervalles de temps non convexes (ou unions finies d'intervalles) [15]. Cela peut être intéressant pour représenter des processus se déroulant en plusieurs phases. Finalement on peut également combiner des modèles qualitatifs et métriques du temps (voir [12]).

2.3 Exemple d'utilisation

Les relations entre intervalles de Allen peuvent être utilisées pour représenter une succession d'événements sous forme d'un réseau de contraintes [3], comme décrit dans la figure 2(a). L'énoncé « Alfred lisait son journal

[1] L'égalité selon Allen entre deux intervalles implique l'égalité de leur durée.



(a) Graphe d'origine (b) Propagation des contraintes

FIG. 2 – Représentation des événements précédant le départ d'Alfred en promenade.

tout en prenant son petit déjeuner. Il acheva sa tasse de café et posa son journal. Après le petit déjeuner, il partit se promener » est représenté au moyen de quatre intervalles de temps, associés à des actions : lire son journal, prendre son petit déjeuner, boire son café, partir se promener. Les relations temporelles entre ces événements sont représentées par des disjonctions de relations de Allen. Ainsi « Après le petit déjeuner, il partit se promener » devient $I_d\ \{m, b\}\ I_p$ représenté par une arête entre les nœuds I_d (ou `Petit-Déjeuner`) et I_p (ou `Promenade`) du graphe. L'arête est étiquetée par la relation $\{m, b\}$ qui signifie *juste avant ou avant*. On pourrait également afficher la relation inverse $\{mi, a\}$ entre I_p et I_d. De la même manière, l'énoncé « Alfred lisait son journal tout en prenant son petit déjeuner » est représentée par l'arête $\{d, di, o, oi, f, fi, s, si, eq\}$ entre les nœuds I_j (ou `Journal`) et I_d : l'information temporelle « tout en » étant très vague, on garde *a priori* toutes les relations incluant un recouvrement des deux intervalles.

L'utilisation des règles de composition des relations de Allen permet de simplifier les étiquettes des relations (par des techniques de propagation de contraintes [7]). Par exemple, sachant que :

(1) la lecture du journal et le café s'achèvent ensemble[2] : $I_j\ \{f, fi\}\ I_c$

(2) le café est une partie du petit déjeuner : $I_c\ \{s, f, d\}\ I_d$

on en conclut que les relations oi, di, si sont impossibles entre I_j et I_d (la lecture du journal ne peut s'achever après le petit déjeuner, figure 2(b)). De fait, par composition :

$$I_j\ \{f, fi\}\ I_c \wedge I_c\ \{s, f, d\}\ I_d \rightarrow I_j\ \{d, o, f, fi, s, eq\}\ I_d$$

3 Représentation du temps dans les recettes de cuisine

3.1 Temps explicite

Prenons un exemple tiré de la base de recettes du premier *Computer Cooking Contest*[3]. C'est une recette de gâteau (ou cake salé) aux courgettes (figure 3). La recette est divisée en deux parties, la première (introduite par l'étiquette <IN>) listant les ingrédients, la deuxième (introduite par l'étiquette <PR>) listant des actions ou opérations. Dans cette deuxième partie, le temps est exprimé de manière qualitative ou quantitative. On remarque tout d'abord que la structure même du texte indique un ordre dans les actions. De plus, des marqueurs temporels sont utilisés pour exprimer par exemple des durées de cuisson : « cuire *pendant* 5 minutes ou *jusqu'à* ce que les courgettes soient tendres ».

La séquence d'actions (deuxième partie de la recette) peut être représentée par le graphe de la figure 4. Le graphe est composé de trois actions principales qui se succèdent linéairement (relation $\{b, m\}$). Chaque action peut être décomposée en un sous-graphe, comme montré pour l'action « cuire les oignons et les courgettes ».

3.2 Temps implicite

Si on regarde plus attentivement la partie « ingrédients » de la recette, on remarque que certains des ingrédients nécessitent une préparation. Ainsi, les courgettes et les oignons sont émincés. Cette préparation doit

[2] Les instants où Alfred finit son café et pose son journal ne sont pas distingués.

[3] www.computercookingcontest.net

```
<RECIPE>
<TI>Zucchini, Chile Corn Bake</TI>
<IN>1 tb Vegetable oil</IN>
<IN>1 lb Zucchini; grated</IN>
<IN>1/2 c Chopped onion</IN>
<IN>3  Eggs</IN>
<IN>3 c Cooked rice</IN>
<IN>7 oz Whole kernel corn (canned) drained</IN>
<IN>8 oz Chopped green chilies</IN>
<IN>2 c Cheddar cheese, grated</IN>
<IN>4 oz Crumbled queso fresco*  OR - very mild feta</IN>
<IN>1 ts Salt</IN>
<IN>Vegetable cooking spray</IN>
<PR>Heat oil in large skillet over medium heat until hot.
Add zucchini and onion; cook uncovered, stirring constantly, for 5 minutes
or until zucchini is soft. Remove from heat; set aside.
[Then] Beat eggs in large bowl. Stir in rice, corn chiles, cheese, zucchini
mixture, and salt.  Mix well together.
[Then] Pour into 13 x 9 x 2-inch baking pan coated with cooking spray.
Bake at 375 degrees 45 to 50 minutes or until knife inserted in center
comes out clean.
</PR>
</RECIPE>
```

FIG. 3 – Une recette de gâteau aux courgettes.

avoir lieu avant les actions décrites dans la seconde partie de la recette. Dans le graphe représenté figure 5(a), nous ajoutons ces préparations et indiquons qu'elles doivent être réalisées avant l'action « ajouter l'oignon et les courgettes » (relation $\{b, m\}$). En revanche seule une connaissance du domaine (il est plus long d'émincer des courgettes que de chauffer de l'huile) nous permet d'établir une relation temporelle entre ces préparations et l'action « chauffer l'huile ». Ainsi, à partir de la relation sur les durées :

$$I_{\text{heat-oil}} \{<\} I_{\text{grat-zucchini}}$$

et des relations déduites du graphe :

(1) $I_{\text{heat-oil}} \{fi\} I_{\text{oil-hot}} \wedge I_{\text{oil-hot}} \{m\} I_{\text{add-zucchini}} \rightarrow I_{\text{heat-oil}} \{m\} I_{\text{add-zucchini}}$

(2) $I_{\text{grat-zucchini}} \{b, m\} I_{\text{add-zucchini}}$

On peut inférer que l'émincage des courgettes doit débuter avant la mise en chauffe de l'huile (figure 5(b)) :

$$I_{\text{grat-zucchini}} \{b, m, o, fi\} I_{\text{heat-oil}}$$

4 Raisonner à partir de recettes

Nous esquissons ici quelques propositions pour une remémoration et une adaptation de recettes de cuisine s'appuyant sur des connaissances temporelles. Cette approche utilise les modèles du temps décrits ci-dessus et nécessite évidemment la mise en œuvre de connaissances du domaine, mais nous n'approfondirons pas ce point.

4.1 Remémoration

La remémoration implantée dans le système TAAABLE [5] est *a priori* indépendante de la représentation temporelle. En effet, le langage des requêtes ne permet pas d'exprimer de contraintes temporelles. Une requête indique quels ingrédients sont souhaités, quels ingrédients sont à éviter, quels types de plats (gâteau, plat

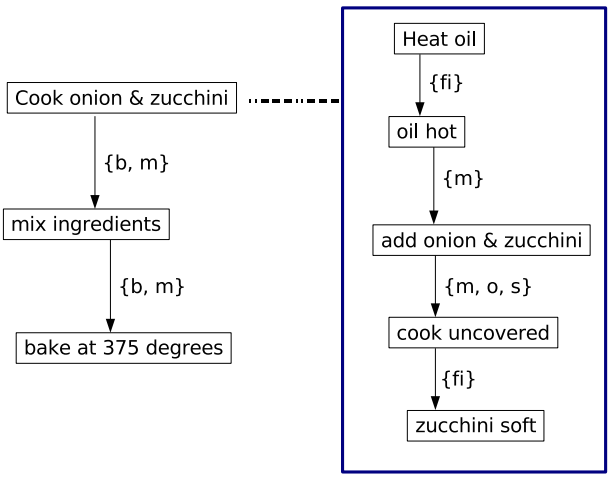


FIG. 4 – Préparation d'un gâteau aux courgettes : graphe d'actions.

chinois, dessert) sont souhaités ou à éviter. Ainsi, la requête $Q = \texttt{gâteau} \wedge \texttt{aubergine} \wedge \neg\texttt{cannelle}$ se lit « Je cherche une recette de gâteau aux aubergines mais sans cannelle. » Étant donné une hiérarchie des ingrédients et des types de plats, TAAABLE va chercher la généralisation minimale Q' de Q telle qu'il existe des recettes satisfaisant la requête Q' [5]. Par exemple, $Q' = \texttt{gâteau} \wedge \texttt{CAeaLdmG} \wedge \neg\texttt{cannelle}$ est une généralisation possible si les concepts « courgette » et « aubergine » sont des sous-concepts de « CAeaLdmG[4] » et si la recette présentée dans la section 3 est dans la base de recettes.

Ainsi, la remémoration de TAAABLE peut rester inchangée même si l'on intègre une représentation temporelle des recettes. En revanche, si on étend le langage des requêtes pour y intégrer des contraintes temporelles, l'algorithme de remémoration devrait intégrer des inférences temporelles de vérification de ces contraintes. Par exemple, on peut désirer qu'une recette demande un temps de préparation limité à 30 minutes et que cette contrainte soit ajoutée à la requête. La remémoration pourrait alors donner une préférence aux recettes respectant cette contrainte.

4.2 Adaptation

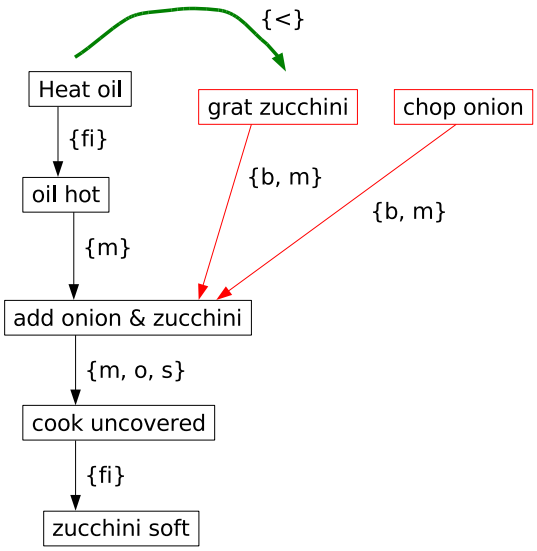
Supposons que l'on dispose d'aubergines et de poivrons. Comme expliqué ci-dessus, à partir de cette liste d'ingrédients, le système TAAABLE peut retrouver la recette du gâteau aux courgettes. Pour adapter cette recette et produire une recette opérationnelle de gâteau aux aubergines et aux poivrons, le système devra proposer une séquence d'actions. Pour cela il devra étiqueter le graphe d'actions avec les nouveaux ingrédients (en remplaçant par exemple les courgettes par les aubergines et les oignons par les poivrons) puis adapter la structure du graphe, c'est-à-dire modifier les étiquettes des relations et éventuellement ajouter ou supprimer des nœuds et des arêtes.

Dans certaines situations l'adaptation est une substitution élémentaire. Par exemple dans le sous-graphe de la figure 4, la partie $\texttt{cuire}\,\{fi\}\,\texttt{courgettes-tendres}$ devient $\texttt{cuire}\,\{fi\}\,\texttt{aubergines-tendres}$: c'est au cuisinier de vérifier que les courgettes ou les aubergines sont devenues tendres. En revanche si l'on s'appuyait sur l'énoncé « cuire 5 minutes » il faudrait utiliser une connaissance du domaine sur les temps de cuisson comparés des aubergines et des courgettes, comme nous le montrons dans le premier exemple ci-dessous.

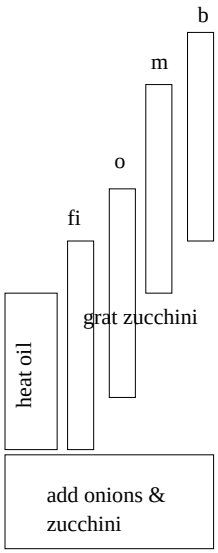
4.3 Exemple 1

Prenons maintenant un exemple de requête $Q = \texttt{bœuf} \wedge \texttt{brocolis}$ ayant conduit à la remémoration d'une recette de bœuf carottes aux haricots verts. Dans la recette originale, les haricots sont ajoutés 10 mn avant la fin

[4]Quand on sait que CAeaLdmG est un acronyme pour « Courgettes, Aubergines et autres Légumes du même Genre » on peut se convaincre de cela.



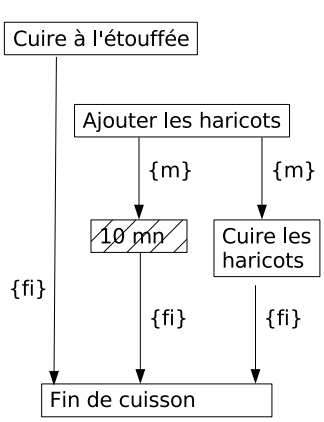
(a) Graphe incluant la préparation des ingrédients : la relation $\{<\}$ entre « heat oil » et « grat zucchini » est issue d'une connaissance du domaine.



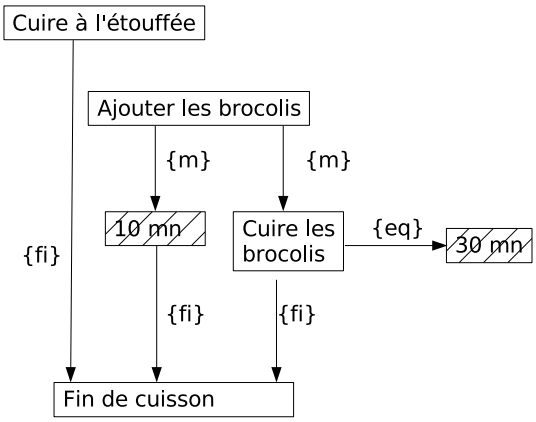
(b) Relations temporelles possibles entre « heat oil » et « grat zucchini ».

FIG. 5 – Préparation d'un gâteau aux courgettes : ajout de relations par utilisation des connaissances du domaine et d'inférences temporelles.

de cuisson (figure 6(a)). Nous introduisons ici des intervalles de durée de référence afin de figurer les informations temporelles quantitatives. Ce graphe est adapté dans un premier temps par une substitution de `brocolis` à `haricots`. Cependant nous disposons d'une connaissance du domaine, établissant que les brocolis doivent cuire 30 minutes. Nous aboutissons ainsi au graphe présenté figure 6(b) qui contient une contradiction : en effet, par inférence, les intervalles-durées I_{10} et I_{30} débutent et finissent ensemble et sont donc égaux (relation $\{eq\}$), alors même que, par définition, I_{10} $\{<\}$ I_{30}.



(a) Préparation du bœuf-haricots



(b) Graphe adapté pour le bœuf-brocolis

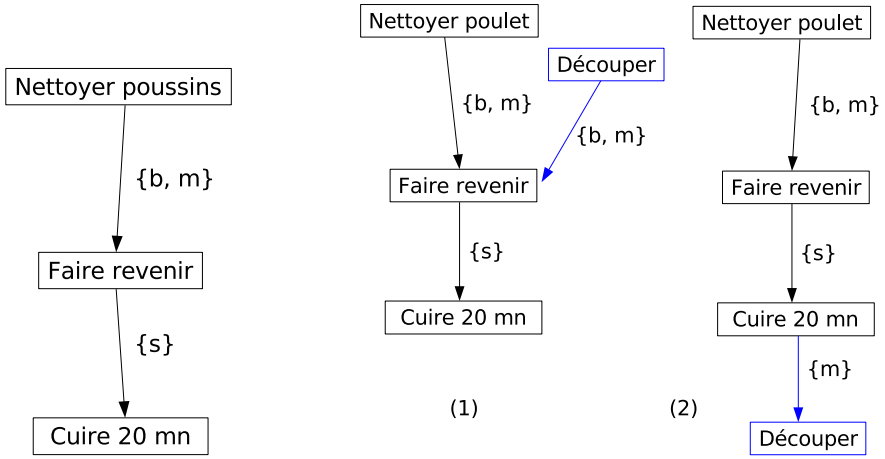
FIG. 6 – Adaptation de recettes avec contrainte sur la durée.

L'adaptation consiste donc dans un premier temps à faire les substitutions correspondant à l'appariement (`haricots` $\rightsquigarrow$ `brocolis`, dans l'exemple) puis à rétablir la cohérence. Pour cela, toutes les informations associées à la recette n'ont pas le même statut : leur statut dépend de leur provenance. Ainsi : (1) les 10 minutes viennent de la recette source, (2) les brocolis viennent de la requête et (3) les 30 minutes viennent des

connaissances du domaine. Les contraintes (2) et (3) doivent être respectées alors que (1) peut être modifiée. Cela peut se ramener à une adaptation conservatrice [6] : on cherche une modification minimale (selon un certain critère) de la recette source qui satisfait à la fois la requête cible et les connaissances du domaine.

4.4 Exemple 2

Nous présentons ici un deuxième exemple, où la requête $Q = \texttt{poulet} \wedge \texttt{amandes}$ renvoie une recette de poussins aux amandes, dont la figure 7(a) donne quelques étapes. L'adaptation consiste en un remplacement des poussins par le poulet, mais, un poulet étant plus gros qu'un poussin, une opération de découpe du poulet doit être introduite dans la recette. Deux possibilités se présentent alors (figure 7(b)) : l'opération de découpe est introduite (1) avant la cuisson ou (2) après la cuisson. Par ailleurs, une connaissance du domaine établit que le temps de cuisson du poulet entier est plus long que celui du poussin et que celui du poulet découpé. Dans la première adaptation envisagée, on pourrait s'arrêter là, tandis que pour la deuxième option, le temps de cuisson est à modifier, comme dans l'exemple précédent.



(a) Préparation des poussins aux amandes.

(b) Graphes adaptés pour le poulet aux amandes.

FIG. 7 – Adaptation de recettes avec ajout d'opération.

4.5 Première synthèse

Suivant ces exemples, trois opérations de vérification des recettes construites peuvent être imaginées en s'appuyant sur le modèle *INDU* évoqué ci-dessus. On distingue en particulier entre « contrainte temporelle globale » associée à la recette – le temps global de préparation par exemple – et « contrainte temporelle locale (ou interne) » qui porte sur les opérations effectuées pendant la recette.

- Si X et Y sont des intervalles de temps, alors Y peut être substitué par X dès que $X \ \{<\} \ Y$: une recette (ou une partie d'une recette) qui nécessite une certaine durée peut être substituée par une recette qui nécessite une durée plus courte si le temps est limité.
- Si une opération op_1 précède nécessairement une opération op_2, $op_1 \ \{b\} \ op_2$, alors op_1 ne peut être substituée que par une opération op_3 de même nature, c'est-à-dire qui respecte la contrainte temporelle $op_3 \ \{b\} \ op_2$. Par exemple, si l'opération « faire revenir les courgettes » précède l'opération « mouiller avec du jus de tomates », alors il doit en être de même pour « faire revenir les aubergines » si « aubergines » devient un substituant de « courgettes ».
- De façon duale, une contrainte sur la précédence peut être considérée comme une contrainte sur la conséquence : l'opération « faire cuire les spaghettis » doit suivre « faire bouillir l'eau dans une grande casserole » tout comme pour « faire cuire les macaronis » si « spaghettis » est un substituant de « macaronis ».

Le raisonnement peut se compliquer si des compositions quelconques de relations temporelles sont considérées, ce qui doit faire l'objet d'un travail de recherche plus conséquent et plus abouti. D'autre part, un travail important de modélisation du domaine doit être réalisé, comme l'ont montré les quelques exemples que nous avons traités.

5 Discussion et conclusion

Différents travaux se sont portés sur la représentation du temps dans le cadre du raisonnement à partir de cas. La plupart d'entre eux s'intéressent à l'analyse ou la prédiction de processus temporels (diagnostic de pannes ou de maladies à partir d'observations régulières ou d'événements successifs, par exemple). L'aspect temporel est pris en compte via l'enchaînement des événements ou éventuellement des dates relatives ou absolues [8, 16, 18]. Seuls les travaux décrits dans [11, 9], à notre connaissance, utilisent une représentation du temps à base d'intervalles et de relations de Allen. Dans [9], les cas sont représentés par des graphes temporels et la remémoration s'appuie sur un appariement de graphes. Dans [11], les cas sont indexés par des chroniques et des contraintes temporelles qui sont décrites par quelques relations de Allen.

D'autres travaux peuvent avoir un intérêt par rapport à notre travail de recherche. Ainsi l'article [4] propose une façon de représenter des plans d'actions dans le cadre d'une logique de descriptions de type $\mathcal{ALC}$ et un des deux exemples principaux porte sur la représentation de recettes de cuisine et l'imbrication temporelle des différentes actions intervenant dans un plan.

Plus généralement, la construction de recettes peut se rattacher également au domaine de la planification : la planification à partir de cas est un champ de recherche du RÀPC qui est bien étudié (voir [20, 13]) et encore actif [19]. Un des systèmes pionniers du RÀPC est un planificateur à partir de cas dans le domaine des recettes de cuisine : CHEF [10]. Les cas de ce système sont des plans représentant des recettes sous la forme d'une séquence d'actions culinaires. Une telle séquence a_1 ; a_2 ; ... ; a_n peut être vue comme un ensemble d'actions a_i et de relations temporelles I_{a_i} $\{b, m\}$ $I_{a_{i+1}}$ ($1 \leqslant i \leqslant n-1$). Sous cet angle, le travail présenté dans cet article peut être considéré comme une extension du langage de représentation des cas (ou plans ou recettes). À l'inverse, les notions de buts et de réalisations de (sous-)buts par des actions manque dans notre modèle alors qu'il est fondamental en planification. Ainsi, l'exemple du bœuf aux brocolis est traité sous l'angle de la résolution de contraintes temporelles à la section 4.3 alors qu'il est traité par CHEF sous l'angle de la réalisation de buts. Dans les deux approches, il s'agit de transformer la solution obtenue par simple substitution (`haricots` $\rightsquigarrow$ `brocolis`) en une solution vérifiant certaines propriétés (contraintes temporelles et réalisation de buts, respectivement). La perspective de recherche qui découle naturellement de cette observation est celle de la combinaison des deux approches.

Références

[1] J. F. Allen. An interval-based representation of temporal knowledge. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81)*, pages 221–226, 1981.

[2] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11) :832–843, 1983.

[3] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2) :123–154, 1984.

[4] A. Artale et E. Franconi. A Temporal Description Logic for Reasoning about Actions and Plans. *Journal of Artificial Intelligence Research*, 9 :463–506, 1998.

[5] F. Badra, R. Bendaoud, R. Bentebitel, P.-A. Champin, J. Cojan, A. Cordier, S. Desprès, S. Jean-Daubias, J. Lieber, T. Meilender, A. Mille, E. Nauer, A. Napoli, et Y. Toussaint. Taaable : Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In *ECCBR Workshops, Workshop of the First Computer Cooking Contest*, pages 219–228, 2008.

[6] J. Cojan et J. Lieber. Conservative Adaptation in Metric Spaces. In *Proceedings of the 9th Conference on Case-Based Reasoning (ECCBR-08)*, LNAI 5239, pages 135–149. Springer, 2008.

[7] J.-F. Condotta et E. Würbel. Réseaux de contraintes temporelles et spatiales. In F. Le Ber, G. Ligozat, et O. Papini, editors, *Raisonnements sur l'espace et le temps : des modèles aux applications*, Traité IGAT - Géomatique, chapitre 7, pages 181–223. Lavoisier, 2007.

[8] M. Dojat, N. Ramaux, et D. Fontaine. Scenario recognition for temporal reasoning in medical domains. *Artificial Intelligence in Medicine*, 14 :139–155, 1998.

[9] M. Dørum Jære, A. Aamodt, et P. Skalle. Representing temporal knowledge for case-based prediction. In *Advances in Case-Based Reasoning, Proceedings of ECCBR 2002*, LNAI 2416, pages 174–188. Springer-Verlag, 2002.

[10] K. J. Hammond. Case-Based Planning : A Framework for Planning from Experience. *Cognitive Science*, 14(3) :385–443, 1990.

[11] M. Jaczynski. Modèle et plate-forme à objets pour l'indexation par situations comportementales : application à la navigation sur le Web. Thèse de doctorat, Université de Nice-Sophia-Antipolis, décembre 1998.

[12] H. A. Kautz et P. B. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In *AAAI-91 Proceedings*, pages 241–246, 1991.

[13] J. Koehler. Planning from Second Principles. *Artificial Intelligence*, 87 :145–186, 1996.

[14] F. Le Ber, G. Ligozat, et O. Papini, editors. *Raisonnements sur l'espace et le temps : des modèles aux applications*. Traité IGAT - Géomatique. Lavoisier, Paris, 2007.

[15] G. Ligozat. Generalized intervals : a guided tour. In *Proceedings of ECAI Workshop on Spatial and Temporal Reasoning, Brighton*, pages 11–18, 1998.

[16] J. Ma et B. Knight. A Framework for Historical Case-Based Reasoning. In *Case-Based Reasoning Research and Development, Proceedings of ICCBR 2003*, LNCS 2689, pages 246–260, 2003.

[17] A.K. Pujari, G.V. Kumari, et A. Sattar. Indu : an interval and duration network. In *Australian Joint Conf. on Artificial Intelligence*, pages 291–303, 1999.

[18] M. Sánchez-Marré, U. Cortés, M. Martínez, J. Comas, et I. Rodríguez-Roda. An approach for temporal case-based reasoning : Episode-based reasoning. In *Proceedings of ICCBR 2005*, LNCS 3620, pages 465–476, 2005.

[19] N. Sugandh, S. Ontañón, et A. Ram. Real-Time Plan Adaptation for Case-Based Planning in Real-Time Strategy Games. In *Advances in Case-Based Reasoning, Proceedings of ECCBR 2008, Trier, Germany*, LNCS 5239, pages 533–547. Springer, 2008.

[20] M. M. Veloso. *Planning and Learning by Analogical Reasoning*. LNAI 886. Springer Verlag, Berlin, 1994.

[21] M. Vilain et H. Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI-86 Proceedings*, pages 377–382, 1986.

Quelles connaissances pour se mettre à TAAABLE ?

Sylvie Despres[1], Haïfa Zargayouna[2], Rim Bentebibel[2]
[1] Laboratoire d'informatique médicale
Bioinformatique (LIM&Bio),
Université Paris 13
UFR de Santé Médecine et Biologie humaine, 74, rue Marcel Cachin 93017 Bobigny France
[2] Laboratoire d'Informatique de l'université Paris-Nord (LIPN) - UMR 7030
Université Paris 13 - CNRS
99, avenue Jean-Baptiste Clément - F-93430 Villetaneuse, France

Résumé

Dans ce papier nous rapportons une expérience prospective d'évaluation dans le cadre du projet TAAABLE 2. Le but du projet de mettre en place un système de raisonnement à partir de cas en ligne pour répondre à des requêtes culinaires. Notre rôle dans ce projet est de préparer une base de test qui permettra au système WIKITAAABLE (un wiki sémantique) d'améliorer ses résultats. Nous nous focalisons essentiellement sur les ressources sémantiques. L'analyse des résultats obtenus met en évidence la nécessité de disposer de ressources adaptées à la tâche de raisonnement (remémoration et adaptation) pour retrouver et faire des propositions de modification des recettes.

Mots clés : RÀPC, adaptation, évaluation, connaissances,

1 Introduction

Dans ce papier nous rapportons une expérience prospective d'évaluation dans le cadre du projet TAAABLE 2. Ce projet fait suite au projet TAAABLE[1] [2] et a pour objectif de mettre en place un système de raisonnement à partir de cas en ligne pour répondre à des requêtes culinaires. Les résultats de ce projet (comme ceux de son prédécesseur) sont soumis à la compétition CCC, *Computer Cooking Contest*[2] organisée dans le cadre de la conférence ICCBR.
A l'instar de l'année dernière, la compétition CCC propose plusieurs tâches : (i) une tâche obligatoire qui est de répondre à une requête impliquant la sélection et la modification d'une recette pour un plat unique[3] ; (ii) une tâche dite d'adaptation qui vise à proposer une adaptation des ingrédients et de préparation pour des recettes de pâtes ; (iii) et enfin une tâche de création de menu est également proposée.

Le système TAAABLE prend en entrée des requêtes et retourne une liste de recettes qui devraient correspondre à ces requêtes. Les critères d'évaluation du concours sont assez divers et sont classés selon trois aspects : (i) la qualité culinaire des recettes créées (créative, appropriée, etc.), (ii) la qualité technique du système (performance, passage à l'échelle, etc.) et (iii) l'originalité scientifique de l'approche (par rapport à une recherche classique, innovation, ...).

Pour notre étude, nous mettons l'accent sur le dernier critère. L'originalité du système est d'intégrer un ensemble de connaissances du domaine dans le processus d'adaptation. Par conséquent, nous centrons notre étude sur les connaissances disponibles dans les ressources du système, leur type ainsi que les mécanismes de raisonnement qui peuvent servir à l'adaptation.

[1] http://taaable.fr/
[2] Computer Cooking Contest, http://www.wi2.uni-trier.de/ccc09/index.php
[3] Seuls les ingrédients figurant dans les recettes de la base peuvent être utilisés.

Notre rôle dans ce projet est de préparer une base de test qui permettra au système WIKITAAABLE [4] d'améliorer ses résultats. Cette étude nous a permis de dégager les questions suivantes, auxquelles nous répondons dans ce papier :

– Comment déterminer les connaissances utiles au système pour répondre aux requêtes de la compétition ?
– Comment évaluer l'apport de l'ontologie au RÀPC plus particulièrement dans la phase d'adaptation ?

L'évaluation d'ontologies est devenue une question cruciale pour ceux qui travaillent à la construction de ces ressources ainsi que ceux qui les utilisent. L'objectif d'avoir un cadre méthodique et des guides pour instrumenter de telles évaluations est assez récent [3]. A notre connaissance, il n'y a pas eu de travaux qui décrivent une expérience d'exploration manuelle telle que celle que nous décrivons dans cet article, ce qui explique le manque de références bibliographiques.

2 Le contexte de l'étude

2.1 La compétition CCC

Le projet TAAABLE participe à la compétition scientifique internationale Computer Cooking Contest (CCC). L'idée de cette compétition est de faire s'affronter des systèmes informatiques capables de résoudre des problèmes de cuisine.

Nous nous sommes intéressées dans un premier temps à la tâche obligatoire de la compétition. Selon les règles du concours, les systèmes candidats devaient être à même de répondre à cinq requêtes en guise d'exercice pour tester les systèmes en retrouvant et/ou en adaptant des recettes existantes afin qu'elles satisfassent les contraintes imposées par les utilisateurs. Ces requêtes exercice sont représentatives des requêtes qui peuvent être posées dans le cadre du concours, elles portent sur : (i) les ingrédients à inclure ou éviter, (ii) le type d'ingrédients (viande, légumes, etc.), (iii) les pratiques alimentaires (régime de cholestérol, régime de goutte, aliments de saison), (iv) type de repas (entrée, dessert, soupe, etc.) et le type de cuisine (italien, asiatique, etc.). Les requêtes sont écrites en langue naturelle, un *main focus*[4] peut être spécifié. La liste des requêtes est présentée ci-dessous.

REQ1 : Cook an Asian soup with leek.[5]
(Main focus : type of meal, type of cuisine)
REQ2 : I would like to have a salad with celery. Please consider that I follow a gout diet.[6]
(Main focus : dietary practice)
REQ3 : Prepare a low-cholesterol dessert with strawberries and avoid citrus fruits.[7]
(Main focus : dietary practice)
REQ4 : Cook a risotto with carrots.[8]
(Main focus : similarity / modification of recipes)
REQ5 : I would like to cook a pear pancake.[9]
(Main focus : similarity / modification of recipes)

La compétition fournit également :

– La base de recettes : les recettes sont des documents textuels en XML faiblement structurés (cf. un exemple de recette présenté en annexe) par les éléments suivants : titre de la recette (<TI>), liste d'ingrédients (<IN>) et préparation (<PR>). Les recettes comportent par ailleurs d'autres éléments qui ne sont pas identifiés explicitement, tels que les ustensiles et les quantités d'ingrédients.

[4]Cette notion de *main focus* n'est pas définie explicitement dans la compétition, nous discutons cette notion plus loin dans le papier.
[5]REQ1 : Préparez une soupe asiatique avec du poireau
[6]REQ2 : Je voudrais une salade avec du céleri. Veuillez considérer que je suis un régime de goutte.
[7]REQ3 : Préparez un dessert avec des fraises avec un bas taux de cholestérol, évitez des agrumes.
[8]REQ4 : Préparez un risotto avec des carottes.
[9]REQ5 : Je voudrais préparer une crêpe aux poires.

– Une liste d'aliments autorisés ou à proscrire dans le cadre de régime (goutte, cholestérol)

2.2 Les ressources du système TAAABLE

Les différentes ressources intervenant dans le système TAAABLE sont [2] :
– Une ontologie des ingrédients (« Ingredient.owl ») organisée selon des liens hiérarchiques et ne comporte pas de rôle. Elle est utilisée lors de la phase d'adaptation. Ainsi la substitution des ingrédients s'effectue en fonction de leur proximité sémantique dans la hiérarchie (par exemple on substitue les pommes par les poires plutôt que les myrtilles) ;
– Une ontologie de type et origine des recettes (« TypeRecetteEtOrigine.owl ») qui contient les types de plats (dessert, soupe, etc.), les origines des plats (chinois, indien etc.) ;
– Une base terminologique construite à partir du *Cook's Thesaurus*[10], elle permet d'établir les liens entre l'ontologie Ingredient et les occurences d'ingrédients dans la base de recettes ;
– La base de recettes annotées est la base de recettes enrichie à l'aide des concepts de l'ontologie de l'ontologie type et origine des recettes ;
– Une base de règles d'adaptation qui sont appliquées quand le système ne trouve pas de solution à l'aide de l'ontologie. Le système de RÀPC peut compléter ce manque en utilisant une base de règles d'adaptation pour adapter les recettes.

3 Comment déterminer les connaissances utiles au système pour répondre aux requêtes ?

3.1 Le processus d'évaluation manuelle

Afin d'établir les connaissances permettant au système de répondre de façon pertinente aux requêtes, nous procédons en trois étapes :
– construction d'une liste de référence de recettes (*gold standard*), en définissant manuellement les recettes pertinentes par rapport aux requêtes culinaires ;
– caractérisation des ressources nécessaires au système en explicitant notre raisonnement afin de mettre en évidence les connaissances utilisées ;
– évaluation des résultats obtenus par le système relativement au gold standard établi.

La construction du gold standard a été réalisée pour chacune des requêtes à partir du traitement de l'ensemble des recettes annotées . Dans le contexte de la compétition, une requête est constituée d'une liste de termes et comporte également une indication sur les éléments qui sont à privilégier pour la recherche. Cette indication est le *main focus* (MF) qui porte sur : (i) type plat combiné avec des ingrédients ; (ii) type de cuisine combiné avec des ingrédients ; (iii) régime ; (iv) procédé pour la recherche ; (v) une combinaison des précedents. La recherche des recettes candidates porte sur le titre de la recette, l'annotation, les ingrédients et la préparation. La sélection des recettes jugées pertinentes parmi l'ensemble de recettes candidates est guidée par les connaissances expertes du domaine de la cuisine. Ces connaissances expertes proviennent à la fois de nos propres expériences et des connaissances disponibles sur des sites relatifs à la cuisine, aux régimes alimentaires, à la diététique et aux encyclopédies dans lesquelles des classifications sont fournies (cf. annexe).

La pertinence d'une recette relativement à une requête est calculée selon une échelle de jugement graduée de 0 à 4. Il est en effet nécessaire de distinguer les réponses correspondant totalement à la requête (recette totalement pertinente (TP)) de celles correspondant partiellement à la requête (recette partiellement pertinente (PP)). Par exemple, si la recherche porte sur une « tarte aux myrtilles », la question est comment juger de la pertinence des réponses « tarte aux fraises » et « confiture aux myrtilles ».

La mesure de pertinence qui permet d'évaluer l'adéquation de la recette sélectionnée avec la requête prend en compte l'importance des éléments contenus dans la requête et le MF. Nous distin-

[10] http://www.foodsubs.com/

guons deux types d'éléments dans la requête : élément central (EC) et élément périphérique (EP) (voir tableau 1). Les substitutions seront d'abord effectuées sur les éléments périphériques. Cette mesure est établie à partir de la comparaison entre la recette candidate et la traduction de la requête en terme d'EC et EP. Les actions qui ont conduit à la sélection de la recette peuvent se résumer par : (i) aucune action car identité ; (ii) spécialisation de l'un des éléments (EC et/ou EP) ; (iii) généralisation de l'un des éléments (EC et/ou EP) ; modification de de l'un des EP (remplacement par un EP équivalent, ajout ou retrait d'un EP). Ces actions sont traduites par la mesure de pertinence ainsi établie.

EC	EP	Pertinence
N	N	0
N	PP ou TP	0
PP	N	1
PP	PP ou TP	2
TP	N	2
TP	PP	3
TP	TP	4

Tab. 1 – Échelle graduée de pertinence. N : non pertinent, PP : Partiellement Pertinent, TP : Totalement Pertinent

La principale difficulté de ce processus de recherche est le repérage de l'EC. Ainsi s'il est relativement naturel dans la requête 1 « Asian soup with leek » avec pour MF « type of meal et type of cuisine » de définir l'EC comme « Asian soup », il est moins facile de le déterminer pour les autres requêtes. Une première heuristique pour retrouver automatiquement l'EC et les EP associés à ce type de requête serait de considérer les éléments apparaissant après le « with » dans le MF.

Les requêtes 2 et 3 ont un MF qui porte sur des régimes alimentaires aux caractéristiques différentes. Le « gout diet » définit des catégories d'aliments à éliminer de l'alimentation tandis que le « low cholesterol diet » est relatif à des quantités acceptables de catégories d'aliments dont la consommation est à réguler. Dans ces deux cas, l'EC est fourni dans la requête. On cherche par conséquent des recettes correspondant à l'intitulé de la requête et respectant les contraintes du régime alimentaire. L'EC associé à la requête 3 est « dessert », les EP sont « strawberries ». Une première recherche a porté sur ces éléments. Une fois l'ensemble des recettes candidates constituées une recherche sur les éléments à éviter pour le régime alimentaire indiqué et la présence de « citrus fruits » permet d'éliminer les recettes non pertinentes. Il est à noter que la formulation « avoid citrus fruits » pourrait laisser la possibilité de conserver des recettes contenant des « citrus fruits ». A terme, la mesure de pertinence devrait pouvoir prendre en compte cette nuance. La seconde recherche associée à cette requête a pour EC « strawberries » dans la mesure où « dessert » est très générique et par conséquent très lié à l'annotation de la recette. Cette stratégie nous a permis de sélectionner une recette TP qui n'était pas annotée.

Enfin pour les requêtes 4 et 5 le procédé indiqué autorise la modification de la recette. Dans ce cas la définition de l'EC devient plus complexe. En effet l'EC peut être un type de plat correspondant à un mode de préparation particulier (risotto with carrots, pear pancake). Dans ce cas, il convient de déterminer ce qui le caractérise et ce qu'il peut supporter comme modification pour être conforme à la recette initiale. Ainsi pour la requête 5 « pear pancake », les recettes de « waffles » peuvent être jugées pertinentes car la pâte utilisée dans les deux préparations est identique et seule la forme change. On peut alors modifier l'EP dans la mesure où la préparation de base est conservée. Pour la recette 4 « risotto with carrots » on peut imaginer généraliser en « rice dish » mais il convient alors de s'assurer que la spécificité de la recette est conservée. La qualité du riz est importante et le principe d'une cuisson par adjonctions successives de petites quantités de bouillon chaud doit être respecté. Il faut par conséquent considérer la préparation de la recette pour décider des variations acceptables. Dans une analyse plus fine, les ustensiles utilisés pour confectionner une recette pourraient constituer des

éléments de sélection ou de rejet. C'est par exemple le cas du « crock pot »pour une distinction entre les soupes et les ragoûts.

3.2 Les résultats de l'exploration manuelle

REQ1 : Cook an Asian soup with leek.

Recette (id : titre)	Pert.	éléments explicatifs
1361 : *Thai Spiced Mussel Soup with Leeks And Carrot Spaghetti*	4	Spécialisation de Asian en Thai.
335 : *Clear Spinach Soup (Korean Malgun Sigumchi Kuk)*	4	Spécialisation de Asian en Korean. De plus on a comme connaissance sur le domaine que scallion peut être substitué à leek ce qui conduit à une nouvelle recherche sur scallion.
775 : *Ken's Hot-And-Sour Soup*	4	Seule soupe apparaît dans le titre, l'annotation « asian dish » permet de sélectionner la recette comme candidate. Ensuite la liste des ingrédients est parcourue et on a comme connaissance sur le domaine que scallion peut être substitué à leek. Dans ce cas l'annotation permet la sélection de cette recette. On aurait pu la retrouver en examinant les ingrédients.

TAB. 2 – Résultat de l'évaluation pour la requête 1.

REQ2 : I would like to have a salad with celery. Please consider that I follow a gout diet.

Recette (id : titre)	Pert.	éléments explicatifs
52 : *Auntie's Luau Potato and Macaroni Salad (Adapted)*	4	« celery » mentionné et « frozen peas » autorisé
348 : *Cold Vegetable Salad*	4	« celery » mentionné « green peas », « green beans » et « green sprouts » autorisés.
412 : *Creamy Waldorf Salad*	4	« celery » mentionné et « sour cream » est un milk product autorisé.
437 : *Curried Rice Salad (K-Parve)*	4	« celery » mentionné et l'ensemble des ingrédients est compatible avec le gout diet.
469 : *Dilly Potato Salad*	4/0	« celery » mentionné, un doute difficile à lever subsiste sur la nature frais ou sec des « beans ». Drained laisse penser qu'il s'agit de fresh peas.
734 : *Iowa Peas and Cheese Salad*	4	« celery » mentionné, cooked green peas et l'ensemble des ingrédients est compatible avec le gout diet.
926 : *New Potato Salad with Dried Tomatoes*	4	« celery » mentionné, low fat milk autorisé.
1241 : *Sky-High Rice Salad*	4	« celery » mentionné et l'ensemble des ingrédients est compatible avec le gout diet.
1299 : *Sprouting Seeds with Carrot Salad and Cottage Cheese*	4	« celery » mentionné, mung bean sprouts autorisés.
1367 : *Thousand-Island Potato Salad*	4	« celery » mentionné et l'ensemble des ingrédients est compatible avec le gout diet.
1383 : *Tomato Vegetable Aspic*	4/0	« celery » mentionné et l'ensemble des ingrédients est compatible avec le gout diet. Mais un aspic peut-il être considéré comme une salade ?
1397 : *Tuna-Waldorf Salad Mold*	4	« celery » mentionné et l'ensemble des ingrédients est compatible avec le gout diet.

Tab. 3 – Résultat de l'évaluation pour la requête 2.

REQ3 : Prepare a low-cholesterol dessert with strawberries and avoid citrus fruits.

Recette (id : titre)	Pert.	éléments explicatifs
320 : *Chocolate-Dipped Strawberries*	4/0	TP si semisweet chocolate peut être retenu
920 : *My Strawberry Pie*	4	l'ensemble des ingrédients est compatible avec les contraintes
588 : *Fruit Cup #1*	4	l'ensemble des ingrédients est compatible avec les contraintes

Tab. 4 – Résultat de l'évaluation pour la requête 3.

REQ4 : Cook a risotto with carrots.

Recette (id : titre)	Pert.	éléments explicatifs
513 : *Eight-Herb Risotto*	3	Ajout de carottes
1157 - 1158 *Risotto Di Zucca* *Risotto with Prociutto and Lemon*	3	Ajout de carottes au moment de la préparation des onions.

Tab. 5 – Résultat de l'évaluation pour la requête 4.

REQ5 : I would like to cook a pear pancake.

Recette (id : titre)	Pert.	éléments explicatifs
32 - 64 *Apple Pancakes From the Townships* *Baked Apple Pancake*	3	Remplace pomme par poire
148 - 1154 *Blueberry Yogurt Pancakes* *Ricotta Filled Crepes with Berry Sauce*	3	Remplace les « berries » par poire
725 : *In-A-Pinch Potato Pancakes*	3/4	Remplace pomme par poire mais un problème de compatibilité avec la recette initiale est posé car la poire est plus sucrée que la pomme. On peut alors ajouter du citron. [a]
1439 : *Very Berry Waffles*	3	Il s'agit de waffle (même préparation) et remplace les « berries » par poire

TAB. 6 – Résultat de l'évaluation manuelle pour la requête 5.

[a]Cette question est évoquée pour cette recette parce que c'est une recette salée, elle ne se pose pas vraiment pour les recettes sucrées.

3.3 Les enseignements à tirer de l'exploration manuelle

Les différentes opérations effectuées pour retrouver les recettes pertinentes sont les remplacements d'un des termes de la requête ou des spécialisation/généralisation ou par des ajouts/retraits d'ingrédients[11] dans la recette :

- remplacement d'un élément de la requête par une opération de :
  - spécialisation : un terme[12] est remplacé par un terme plus spécifique. Par exemple, *soupe asiatique* est remplacé par *soupe chinoise*. La réponse est donc totalement pertinente car elle répond tout à fait à la requête.
  - généralisation : un terme est remplacé par un terme plus général. La réponse peut être considéré comme pertinente si elle ne s'éloigne pas du résultat attendu. Par exemple, remplacer *plat marocain* par *plat maghrébin* est plus pertinent que le remplacer par un plat africain.
  - substitution
    - synonymie ou proximité sémantique : un terme est remplacé par un terme équivalent. Dans le cas des ingrédients, cela peut être un ingrédient avec les mêmes caractéristiques nutritionnelles, mêmes saveurs ou même consistence. Par exemple remplacer *pêche* par *abricot*[13].
    - antonymie : un ingrédient est par exemple remplacé par un autre mais qui lui est contraire. Par exemple, *lait à teneur élevée en matière grasse* peut être remplacé par *lait allégé*.
- remplacement d'un élément de la recette par une opération de :
  - ajout d'un ingrédient : en l'absence de réponse même par modification de la requête, on peut proposer de modifier les recettes en ajoutant des ingrédients. Par exemple, on peut proposer d'ajouter des *carottes* à une recette de *risotto aux herbes*.
  - retrait d'un ingrédient : en l'absence de réponse même par modification de la requête, on peut proposer de modifier les recettes en retirant des ingrédients. Par exemple, on peut proposer

[11]Ce type d'opérations est appliqué dans un premier temps seulement aux ingrédients, il peut dans un second temps être étendu à la préparation.

[12]L'utilisation du terme *terme* à la place de *concept* est motivée par le point de vue usager de l'évaluateur qui n'est pas censé cerner la différence qu'il y a entre ces deux notions

[13]Les auteurs se sont référés au Cooking Dictionary - Ingredients Substitutions and Equivalents accessible via `http://www.gourmetsleuth.com/equivalents_substitutions.asp?` et déclinent donc toute responsabilité.

de retirer *citrus fruit* à une recette pour éviter les agrumes.

Il est à noter que la recherche des recettes candidates peut être perturbée par des phénomènes d'homonymie entre ingrédients. La quantité pour l'ingrédient considéré, l'association d'un adjectif ou l'apposition de deux ingrédients peuvent aider à lever l'ambiguïté. L'exemple de pepper (poivron ou poivre) illustre cette situation : la quantité indique qu'il s'agit plutôt de poivre ou d'un poivron, l'association « green pepper » précise qu'il s'agit d'un poivron, l'apposition « salt and pepper » conduit à supposer qu'il s'agit de poivre.

L'utilisation de l'une ou l'autre de ces opérations est fondée sur des connaissances relatives au domaine de la cuisine et de la diététique. Comme nous l'avons précédemment souligné, ces connaissances peuvent être celles de nos expériences respectives mais également celles disponibles dans des ressources encyclopédiques (classification botanique), médicales (connaissances de régimes plus approfondies que la liste fournie pour la compétition), diététiques (tableau de composition des ingrédients) ou plus pragmatiques (par exemple, les rubriques « truc et astuces » des sites de cuisine). Le processus manuel de recherche dépend de la qualité et la fiabilité des ressources exploitées. Les annotations associées à chaque recette constituent également des connaissances dans la mesure où elles définissent les classes d'appartenance auxquelles les recettes sont associées. Elles préexistaient à notre travail. Il est clair qu'elles aident à retrouver les recettes quand elles sont adéquates. Il peut arriver qu'elles soient absentes et dans ce cas, la recherche doit se faire sur l'ingrédient (cf. exemple recette 588 Fruit Cup). Actuellement, elles peuvent également être redondantes ce qui conduit à sélectionner des recettes candidates qui ne relèvent pas du sous-ensemble défini par la requête (cf. la recette 286 « Chinese Chicken Salad annotée par « asian dish/chinese dish/salad dish/main dish/meat dish/poultry dish/vegetable dish/low fat dish/low calorie dish/fat free dish/dressing dish »). Le processus de recherche est par conséquent totalement dépendant de la qualité du processus d'annotation.

Le gold standard a été établi à partir de la base de recettes annotées et les cinq requêtes fournies pour la compétition de cette année. L'examen des requêtes proposées l'an dernier montre que le MF est construit à partir des mêmes composants, seule les combinaisons varient. Par conséquent, il nous semble possible de créer de nouvelles requêtes pour enrichir notre gold standard en faisant varier les différents éléments intervenant dans la requête et en combinant les composants du MF. L'évolution du gold standard permettra par la suite de définir des nouveaux jeux de tests pour le système et ainsi de mieux anticiper les nouvelles requêtes proposées lors de la compétition. Un algorithme de génération automatique de requête reste à concevoir en s'appuyant sur la caractérisation des composants de la requête et du MF.

4 Comment évaluer l'apport de l'ontologie au RÀPC

On a montré dans la section 3 que la sélection des recettes repose sur divers types de connaissances que nous avons explicitées. La question maintenant posée est de mesurer le gain obtenu si le système utilise de telles ressources. Dans le premier paragraphe, nous discutons l'intérêt d'un tel choix. Puis nous présentons l'évaluation du système par rapport au gold standard. Ensuite à la lumière de ces premiers résultats, après avoir présenté les ressources actuellement exploitées par le système, nous proposons de les faire évoluer afin d'améliorer les performances du système.

4.1 Le jeu en vaut-il la chandelle ?

Divers modèles de RÀPC proposent d'intégrer les connaissances sémantiques (voir par exemple [6], [5], [1]) avec un certain succès. Néanmoins les évaluations de tels systèmes n'ont pas porté sur l'apport spécifique des ontologies. Même s'il paraît assez intuitif que l'intégration de connaissances sémantiques améliore le résultat des systèmes, les ontologies n'ont pas encore démontré leur efficacité de manière probante. L'expérience des systèmes de Recheche d'Information montre des résultats mitigés. Une ressource mal adaptée génère du bruit et détériore les performances du système, de

même une ressource incomplète provoque du silence. L'apport d'une ressource sémantique dépend fortement de la manière dont elle est utilisée, du type d'information nécessaire et de sa qualité [8] [9]. Nous jugeons nécessaire d'étudier ces divers points afin de démontrer de l'utilité de l'utilisation des ontologies. Il est à noter que dans le cadre de la compétition CCC 2008, le système TAAABLE était le seul à intégrer une ontologie dans son raisonnement.

4.2 Évaluation par rapport au gold standard

La comparaison des recettes trouvées manuellement par rapport à celles retrouvées par le système permet de calculer le rappel et la précision [14] en utilisant la pertinence graduée. Deux types de valeurs sont présentées dans le tableau 7 : en tenant compte des recettes ayant obtenu au moins 3 comme note de pertinence [15]. Les mesures de précision et de rappel donnent une idée sur la performance du système mais restent insuffisantes si elles ne sont pas accompagnées d'étude qualitative pour expliquer le bruit et le silence traduit par ces mesures. Nous avons présenté plus haut une synthèse des principales différences, une étude détaillée par recette est présentée en annexe.

	Précision	Rappel
REQ 1.	1	0.33
REQ 2.	0.46	0.5
REQ 3.	0.25	0.66
REQ 4.	0	0
REQ 5.	1	0.6
Total	0.48	0.46

TAB. 7 – Rappel et Précision sur les soumissions préliminaires de TAAABLE aux requêtes de test

Nous comparons les résultats obtenus par le système avec ceux sélectionnés comme pertinents par l'utilisateur. Deux types de différences ont été identifiées :

- l'utilisateur a sélectionné des recettes supplémentaires car il dispose de connaissances plus nuancées que le système ou il est mieux informé ;
- le système a sélectionné des recettes différentes de l'utilisateur car les ressources dont il dispose :
  - comportent des imprécisions. C'est notamment le cas pour le « gout diet » où les « pulse » sont à exclure tandis que les « peas, bean » frais sont autorisés.
  - une contrainte de la requête n'est pas prise en compte. C'est le cas pour les agrumes qui n'ont pas été pris en compte par le système.
  - Le remplacement d'un ingrédient est décidé par l'utilisateur même s'il n'est pas immédiat. On peut remplacer les berries par des poires sans dénaturer le pancake.
  - On peut aussi remplacer par un mode de préparation équivalent. Ainsi pancake peut être remplacé par waffle.

4.3 Évolution des ressources disponibles

Les ressources actuellement disponibles sont la hiérarchie des ingrédients « Ingredient.owl » et la hiérarchie décrivant les types et origines des recettes TypeRecettesEtOrigine.owl. Au vu des résultats présentés dans la section 3, deux types d'opérations (classification et substitution par des éléments

[14] Rappellons que :

$$precision = \frac{|S \cap R|}{|S|} \quad rappel = \frac{|S \cap R|}{|R|}$$

où $|S \cap R|$ est le nombre d'éléments pertinents retournés par le système, $|S|$ est le nombre d'éléments retournés par le système et $|R|$ le nombre d'éléments dans la référence.

[15] Les recettes qui ont obtenu la note 4/0 sont celles où subsiste un doute, elles seront prises en compte en faveur du système *c-à-d* si le système ne retrouve pas, ce n'est pas comptabilisé comme silence.

équivalents) se dégagent pour répondre de façon la plus pertinente possible à la requête. Nous avons par conséquent étudié les ressources disponibles en fonction des possibilités qu'elles offrent au système pour accomplir ces deux actions.

L'ontologie des ingrédients (cf. figure 1) est constituée de 4430 classes primitives (c'est-à-dire dont la définition est exprimée en termes de conditions nécessaires (CN)), non disjointes et organisées dans une hiérarchie. L'ontologie ne comporte ni propriété caractérisant les classes, ni relation entre les classes.

Fig. 1 – Extrait de Ingredient.owl

Fig. 2 – fresh et dried « peas ou beans »

Les classes ne sont pas disjointes et peuvent apparaître à plusieurs endroits dans la hiérarchie. C'est par exemple le cas pour la classe « bean » qui est à la fois sous-classe de « legume » et de « not_gout_diet_compliant » (cf. figure 2). Sur ce même exemple, il apparaît que « bean », « dried bean » et « fresh bean » sont au même niveau hiérarchique. La classe « pea » est définie comme sous-classe de « vegetarian » mais n'est pas liée avec « fresh_pea » et « dried_pea ». La création de classe définie (c'est-à-dire dont la définition est exprimée en termes de conditions nécessaire et suffisante (CNS)) permettrait par exemple de définir la classe « pulse » utilisée pour caractériser les ingrédients non autorisés dans le régime « gout diet ». La définition de cette classe est « The dried seed contained within the pod of several varieties of legumes, which include beans, lentils, and peas. Pulses are low in sodium and calories, while high in complex carbohydrates, fiber and protein. L'existence de cette classe aurait permis au système d'éviter l'erreur correspondant au retour de recette « gout diet » en ne tenant pas compte que les fresh < peas ou beans > sont autorisés tandis que les dried < peas ou beans > ne le sont pas. Cette première analyse fait apparaître la nécessité de retravailler cette ontologie selon les principes prônés par [7] afin de pouvoir raisonner de façon consistante sur la ressource. Les opérations de spécialisation et de généralisation seraient ainsi facilitées.

La qualité des résultats obtenus associés aux opérations de substitution n'est pas optimale car la ressource ingrédients ne comporte pas de connaissances permettant par exemple, de décider de l'équivalence d'ingrédients en tenant compte de leurs propriétés. Dans le processus manuel, nous avons utilisé des ressources externes (cf. annexe) pour pouvoir proposer une substitution d'ingrédients. La classification botanique des ingrédients peut également constituer une aide. Ainsi les « drupéoles » constitue la catégorie des fruits à pépins dans lesquelles on pourra retrouver la fraise, la framboise, la mûre (de la ronce), la pomme et la poire ce qui permet de proposer la substitution de « strawberries » à « pear ». Les trucs et astuces sont fondés sur une classification des caractéristiques chimiques des ingrédients (cf. annexe). Il serait utile de travailler sur cette ressource afin de fabriquer des classes d'ingrédients qui nous permettraient de déduire l'équivalence. Cette ressource permet également de donner des éléments pouvant justifier la substitution d'un ingrédient par plusieurs ingrédients

pour compenser une propriété manquante par exemple en substituant la rhubarbe avec du sucre à la pomme afin d'adoucir l'acidité de la rhubarbe.

L'ontologie des types et origines des recettes (cf. figure 3) est constituée de 160 classes primitives, non disjointes et organisées dans une hiérarchie. L'ontologie ne comporte ni propriété caractérisant les classes, ni relation entre les classes.

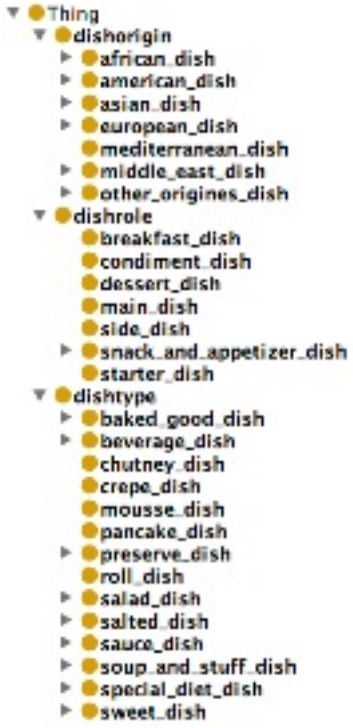


FIG. 3 – Les trois grandes classes de TypeRecetteEtOrigine

Cette ressource constitue un outil pour l'annotation des recettes. Une évaluation de la couverture de cette ressource serait nécessaire afin de garantir la qualité de l'annotation. Il apparaît en effet que certaines recettes sont annotées de façon redondante tandis que d'autres ne sont pas annotées. Une vérification manuelle de l'ensemble des recettes annotées reste à faire afin de détecter les éventuelles erreurs. Elle permettra également d'avoir une idée de la couverture de cette ontologie.

La ressource TypeRecettesEtOrigine.owl pourrait également intervenir dans le processus d'interrogation pour classer une recette. En effet, lors du processus manuel on peut envisager utiliser l'ontologie pour classer une recette proposée par le système et examiner si elle relève bien de la catégorie attendue. Un tel usage suppose que des rôles soient établis entre les deux ressources Ingrédient.owl et TypeRecettesEtOrigine.owl.

Nous avons fait évoluer cette ressource en définissant une classe relative à la notion de « Recipe » (cf. figures 6 et 7) et en créant des relations (cf. figure 4) entre les classes « Recipe » et « DishCategory », « DishOrigin » et « DishType ». Dans la mesure où une recette peut relever de plusieurs types de plat. Ainsi, une salade peut être un StarterDish, un MainDish ou un DessertDish.

Nous avons modifié l'intitulé et le contenu de la classe DishRole afin de mieux distinguer les éléments qui composaient DishRole et DishType (cf. Figure 5). La classe DishOrigin a été légèrement réorganisée, mais n'a pas subi de modification quant au contenu. La relation « hasIngredient » a également été construite pour faire le lien avec la ressource Ingrédient. A ce point du travail, il conviendrait de reprendre la partie dite haute de l'ontologie proposée dans [2] pour continuer à faire évoluer de façon cohérentes les ressources disponibles.

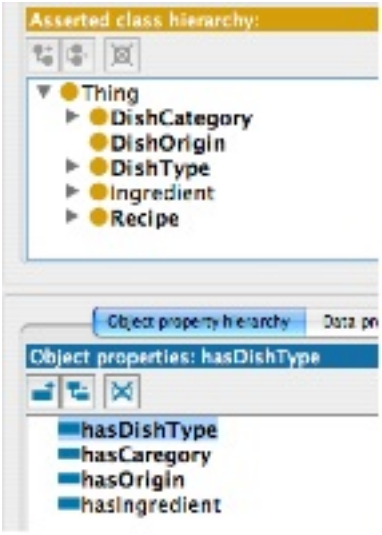


Fig. 4 – Exemple de rôles nécessaires

Fig. 5 – Détail de la classe DishCategory

Fig. 6 – Détail de la classe Recipe

Fig. 7 – Définition de la classe définie RiceRecipe

5 Bilan

L'évaluation des résultats associés aux requêtes lorsque la recherche est effectuée manuellement a suscité de nombreuses discussions. En effet, il a été ardu de déterminer une grille d'évaluation permettant d'obtenir une note consensuelle reflétant la pertinence des recettes retournées. Les raisons de ces difficultés sont en partie dues aux besoins d'expliciter les connaissances utilisées lors du raisonnement mis en œuvre pour sélectionner la recette. L'interprétation de la requête afin de déterminer l'EC constitue une de ces difficultés. La proposition de considérer le mot de liaison « with » comme un indice de détection d'un élément de moindre importance (EP) est issue de ces échanges.
Ensuite, des écueils sont liés à la langue naturelle parmi ceux-ci figurent le phénomène d'homonymie. Les équivalences entre aliment telles que « leek » et « scallion » ou « green onion » ont également un impact dans l'élaboration du consensus autour de la note.
Des discussions ont également porté sur la façon de gérer les régimes qui comme nous l'avons souligné ne sont pas traités de façon équivalente. Il faudrait être capable de prendre en compte les caractéristiques de ces derniers pour retourner des recettes en adéquation avec leurs contraintes respectives. Deux stratégies se dégagent. La première consiste en la suppression des ingrédients à éviter et leur remplacement par des ingrédients équivalents dans le cas des régimes excluant des classes d'aliments. La seconde est le calcul des quantités autorisées quant il s'agit de régime avec des contraintes de quantité de graisse ou de calories à calculer . Certains régimes feront intervenir ces deux stratégies.
La sélection d'une recette peut également se faire sur un mode de préparation. Si l'exemple de « Soup » ne pose *a priori* pas de problème, l'exemple du « risotto » est plus délicat car une sélection de recettes avec une généralisation par « rice dish » est trop imprécise pour obtenir de recettes susceptibles de

convenir. Il est par conséquent utile de définir ces modes de préparation. Un autre avantage d'une telle prise en compte permettrait d'éviter l'erreur du système liée à la préparation « Eagle Brand Milk »[16] qui est constituée d'ingrédients à éviter pour le « low cholesterol diet ». Des doutes subsistent sur certaines recettes en raison d'une méconnaissance des caractéristiques de certains ingrédients ou des points plus subtils tels que l'effet produit par le remplacement d'un ingrédient par un autre. Ainsi les caractéristiques de composition pour « chocolate » pourraient aider à déterminer s'il s'agit d'un aliment autorisé pour le « low cholesterol diet ». De la même manière, il serait intéressant de pouvoir évaluer l'incidence du remplacement de « apple » par « pear » lorsque la pomme intervient dans une recette essentiellement salée.
Les questions suscitées par certaines annotations ou leur absence montrent l'importance qu'elles revêtent dans l'aide au choix. Un travail reste à faire pour juger de leur impact sur la sélection automatique des recettes candidates à une requête donnée. Un premier exemple est celui de la recette 588 « Fruit cup » (cf tableau 5) qui n'est pas annotée et ne contient pas d'indice dans le titre pour déterminer qu'il s'agit d'un dessert contenant des « strawberries ».

Ces différentes constatations et les erreurs répertoriées en terme de bruit et de silence du système lorsqu'elles sont dues à la qualité des ontologies, conduisent à penser qu'en enrichissant ces ressources le raisonnement du système serait facilité. Il faudra toutefois définir des protocoles d'évaluation de ces ressources.
La définition de classes définies des types de plats, des préparations et des régimes devrait améliorer la phase de remémoration des recettes candidates. La difficulté est le traitement de la négation. La solution adoptée consistant à définir les not_diet_complient dans les ressources existantes est efficace à condition que la classe soit définie de façon exhaustive. Ce dernier point pose le problème de la couverture du domaine de la ressource construite ainsi que de sa maintenance. Quelques traces de ces actions d'enrichissement apparaissent avec « bean » dans la ressource TypeRecettesEtOrigineOntologie.
La qualité de ces ressources est également un point crucial pour l'activité d'annotation qui comme nous l'avons déjà souligné est déterminante dans la phase de remémoration.
L'ajout des caractéristiques nutritionnelles des ingrédients, des équivalences de goût entre aliment et des trucs et astuces décrivant des remplacements d'ingrédients *via* des règles d'adaptation pourrait faciliter l'adaptation. La définition de relation (rôle) entre les différentes classes des ontologies disponibles est un élément crucial pour faire des liens au cours des raisonnements.

A l'issue de ce bilan, il est clair que la qualité des ontologies construites détermine la pertinence des résultats du système. En effet la sélection des recettes et leur adaptation par le système sont centrales dans le raisonnement. L'évaluation des ontologies produites est par conséquent incontournable. Lorsque les erreurs du système ne relèvent pas de la qualité de la ressource, l'examen des traces du raisonnement du système (règles d'adaptation utilisées) devrait permettre de les détecter.

6 Conclusion

Cette étude nous a permis de mettre en lumière, l'importance des bases de connaissance pour les systèmes de RÀPC. Ces bases de connaissances, si elles sont bien construites, permettent de mettre en place des raisonnements complexes.
Nous avons montré que des questions récurrentes sont posées : quelles connaissances ? sous quelle forme ? et pour quel type de raisonnement ? Certaines connaissances peuvent être extraites automatiquement du texte, d'autres sont plus difficiles à extraire.
Nous espérons par cette analyse mettre à disposition une base de test qui permettra au système WIKITAAABLE d'améliorer ses résultats et de proposer des recettes créatives. Le *gold standard* que nous avons établi est aussi utile pour les autres systèmes (qui n'utilisent pas de ressources sémantiques).

[16] Le Eagle Brand Milk est une préparation à base de lait, de sucre et de white syrup.

Nous comptons mettre en place des protocoles de test pour les autres tâches de la compétition ainsi que faire émerger des guides de bonne pratique utiles pour l'exploitation de ressources sémantiques..

Remerciements

Nous tenons à remercier Catherine Duclos et Sylvie Salotti avec qui nous avons eu l'occasion de discuter des questions abordées ici, au sein du groupe évaluation du projet TAAABLE..

Nous remercions les relecteurs anonymes pour la qualité de leurs remarques et critiques.

Références

[1] A. Abou Assali, D. Lenne, B. Debray, et S. Bouchet. COBRA : Une plate-forme de RàPC basée sur des ontologies. In Actes des 20es Journées Francophones d'Ingénierie des Connaissances (IC 2009) IC 2009, pages 277–288, Hammamet Tunisie, 05 2009.

[2] F. Badra, R. Bendaoud, R. Bentebibel, P.A. Champin, J. Cojan, A. Cordier, S. Desprès, S. Jean-Daubias, J. Lieber, T. Meilender, A. Mille, E. Nauer, A. Napoli, et Y. Toussaint. TAAABLE : Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In M. Schaaf, editor, ECCBR 2008, The 9th European Conference on Case-Based Reasoning, Trier, Germany, September 1-4, 2008, Workshop Proceedings, pages 219–228, 2008.

[3] A. Baneyx et J. Charlet. Évaluation, évolution et maintenance d'une ontologie en médecine : état des lieux et expérimentations. Revue I3 - Information, Interaction, Intelligence, numéro spécial « Corpus et ontologies », pages 147–173, 2007.

[4] A. Cordier, J. Lieber, P. Molli, E. Nauer, H. Skaf-Molli, et Y. Toussaint. WIKITAAABLE, un wiki sémantique utilisé comme un tableau noir dans un système de raisonnement à partir de cas textuel. In 17ème atelier de Raisonnement à Partir de Cas - 2009 (soumis), 2009.

[5] B. Díaz-Agudo, P. A. González-Calero, J. A. Recio-García, et A. A. Sánchez-Ruiz-Granados. Building cbr systems with jcolibri. Sci. Comput. Program., 69(1-3) :68–75, 2007.

[6] Béatrice Fuchs et Alain Mille. Une modélisation au niveau connaissance du raisonnement à partir de cas, June 2005. Ingénierie Des Connaissances Coordonné par R. Teulier, J. Charlet, P. Tchounikine éditeur L'Harmattan.

[7] N. Guarino et C. Welty. Evaluating ontological decisions with ontoclean. In Communications of the ACM, volume 45, 2002.

[8] F. Moreau, V. Claveau, et P. Sébillot. Intégrer plus de connaissances linguistiques en recherche d'information peut-il augmenter les performances des systèmes ? In Actes de la Conférence CORIA'2007.

[9] H. Zargayouna. Indexation sémantique de documents XML. PhD thesis, Université Paris-Sud, 2005.

Annexe

Exemple de recette

```
<RECIPE>
<ID>1361</ID>
<TI>Thai Pumpkin Soup</TI>
<IN>1 Onion (5-6 oz); finely chopped</IN>
<IN>1/4 c Diced red bell pepper</IN>
```

```
<IN>2 tb Curry powder</IN>
<IN>1 ts Minced garlic</IN>
<IN>1 cn (1-lb) pumpkin</IN>
<IN>2 Chicken bouillon cubes</IN>
<IN>1 ts Sugar</IN>
<IN>2 c Unflavored nonfat yogurt</IN>
<IN>1 tb All-purpose flour</IN>
<IN>1/4 c Chopped fresh cilantro</IN>
<PR>
Notes: If making soup ahead, cool, cover, and chill up to 2 days,
then reheat before serving.
1. Place onion, bell pepper, and 1/2 cup water in a 3- to 4-quart
pan over high heat. Stir often until pan is dry and onion browns
lightly, 10 to 12 minutes.
2. Add curry powder and garlic; stir 1 minute. Add 2 cups water,
stirring well to free browned bits in pan, then add pumpkin,
bouillon cubes, and sugar. Cover and simmer gently to blend flavors,
25 to 30 minutes.
3. Meanwhile, mix a little of the yogurt with the flour until smooth,
then combine with remaining yogurt. When the soup has simmered, with
a whisk, stir yogurt-flour mixture into it. Turn heat to high and
stir until boiling, then simmer and stir about 2 minutes. Ladle into
bowls or mugs, and top with chopped cilantro.
</PR>
</RECIPE>
```

Nous avons ajouté incrémentalement les numéros de recettes délimités par la balise <ID>

Tableau récapitulatif des premiers résultats des systèmes par rapport à l'évaluation manuelle

Nous présentons dans ce qui suit les tableaux comparatifs qui ne présentent que les recettes qui présentent un bruit (retrouvée par le système et rejetée manuellement) ou un silence (retrouvée manuellement mais non présentée par le système).

Recette (id : titre)	Silence/Bruit	Traces de raisonnement
335 : *Clear Spinach Soup (Korean Malgun Sigumchi Kuk)*	Silence	spécialisation asian/korean et équivalence leek/scallion
775 : *Ken's Hot-And-Sour Soup*	Silence	les ingrédients permettent de déterminer qu'il s'agit d'une asian soup et équivalence leek/scallion

Tab. 8 – Évaluation comparative pour la requête 1.

Le système ne retrouve pas les recettes 335 et 775. Une des hypothèses pour expliquer ce silence est qu'une fois retrouvée la recette 1361 qui ne demande qu'une spécialisation le système s'arrête. Parmi les autres éléments explicatifs, le système ne dispose de l'équivalence entre leek et scallion.

REQ2 : I would like to have a salad with celery. Please consider that I follow a gout diet.

Recette (id : titre)	Silence/Bruit	Traces de raisonnement
46 : *Asian Chicken Salad W/spicy Peanut Sauce*	Bruit	chicken appartient à la classe poultry qui n'est pas autorisé dans le gout diet. ⚠Erreur du système

Tab. 9 – Évaluation comparative pour la requête 2.

Le système retourne les recettes 46, 816 et 1029. Elles ne sont pas retenues, car selon notre interprétation du gout diet, elles ne vérifient pas les contraintes imposées par le régime décrit dans le CCC. Chicken appartient à la catégorie des Poultry. Il y a toutefois une vérification à faire du point de vue du régime car si turkey figure dans les listes d'aliments à éviter, chicken n'y est pas forcément.
L'erreur ne vient pas de la ressource Ingredient, car poultry apparaît dans la classe meat qui est un sous classe de not_gout_diet_complient et chicken est une sous classe de poultry.

⚠A noter que le reste des recettes retournées par le système concordent bien avec les recettes retrouvées manuellement. Elles sont néanmoins retournées "grâce" à une erreur dans le système.
Le système retourne les recettes 52, 348, 734, 1299. Ces recettes ne devraient pas être sélectionnées par le système car selon la ressource Ingredient les fresh_bean et fresh_pea font partie des aliments à éviter. **Il y a deux erreurs l'une du système non explicable de façon immédiate et une erreur de modélisation**. Les pulse sont des légumes secs et par conséquent les légumes frais ne sont pas éliminer. **La classe not_gout_diet_complient est à modifier.**

Le système retient 1029 qui comporte à la fois des peas et des beans et du poulet. Deux erreurs sont cumulées mais déjà décrites.

REQ3 : Prepare a low-cholesterol dessert with strawberries and avoid citrus fruits.

Recette (id : titre)	Silence/Bruit	Traces de raisonnement
728	Bruit	Contient du lait, des œufs et de la crème fouettée. ⚠Erreur du système
1315	Bruit	Contient du lait, des œufs et de la crème fouettée. ⚠Erreur du système
1436	Bruit	Nécessité de supprimer orange et lemon juice pour être conforme à la requête. ⚠Erreur du système
1312 : *Strawberry Dip*	Bruit	Contient Eagle Brand Milk qui est fabriqué avec du non fat dry milk, sugar, water and butter.
1313 - 1314 *Strawberry Ice Cream Cake Strawberry Pudding*	Bruit	le Cream Cheese est un aliment à éviter pour un low cholesterol diet.

Tab. 10 – Évaluation comparative pour la requête 3.

Les recettes 320, 588, 920, sont jugées pertinentes.

Le système retourne également les recettes 728, 1312, 1313, 1314, 1315 et 1436. Ces recettes ne devraient pas être sélectionnées par le système.

La recette 728 ne devrait pas être retenue car elle contient du milk. **L'erreur vient de la ressource Ingredient n'est pas considéré comme un not_cholesterol_diet_complient dans la ressource Ingre-**

dient.
La recette 1312 ne peut pas être retenue car elle contient de l'Eagle Brand Milk qui est fabriqué avec Non fat dry milk, sugar, water and butter. Le butter n'est pas un ingrédient autorisé. **L'erreur vient d'un manque de précision de la ressource.**
Les recettes 1313 et 1314 ne peuvent pas être retenues car elles contiennent du cream cheese qui est un aliment à éviter dans le cas d'un pour le low cholesterol diet (`http://www.gicare.com/Diets/low-cholesterol-diet.aspx`). Creamcheese est dans la recette Ingredient comme sous classe de fresh_cheese et cheese n'est pas considéré comme un not_cholesterol_diet_complient dans la ressource Ingredient. **L'erreur du système est liée au contenu de la ressource.**
La recette 1315 ne devrait pas être retenue car elle contient à la fois des eggs et whiping cream qui font partie de la classe not_cholesterol_diet_complient et du milk non référencé not_cholesterol_diet_complient. **L'erreur est sans doute imputable au système.**

Le système ne trouve pas la recette 588. Cette recette ne comporte pas d'annotation. Les ingrédients sont conformes au low cholesterol diet. **L'erreur est sans doute imputable au système mais nous ne disposons d'une explication immédiate.**

REQ4 : Cook a risotto with carrots.

Recette (id : titre)	Pert.	Traces de raisonnement
513 : *Eight-Herb Risotto*	Silence	Ajout de carottes
1158 - 1157 *Risotto with Prociutto and Lemon* *Risotto Di Zucca*	Silence Silence	Ajout de carottes au moment de la préparation des onions.

Tab. 11 – Résultat de l'évaluation pour la requête 4.

Le système ne retrouve aucune des trois recettes. **Ce silence tient sans doute à une erreur de raisonnement du système.** Il n'y a pas de réponse exacte ou proche dans la base de recettes. Nous avons choisi de proposer d'ajouter carottes au recettes (cf. section 3.3). Le système par contre n'en a retourné aucun.

REQ5 : I would like to cook a pear pancake.

Recette (id : titre)	Silence/Bruit	Traces de raisonnement
148 - 1154 *Blueberry Yogurt Pancakes* *Ricotta Filled Crepes with Berry Sauce*	Silence	Modification en remplaçant blueberries par pear
1439 : *Very Berry Waffles*	Silence	Il s'agit de waffle (même préparation) et Remplace les « berries » par poire

Tab. 12 – Évaluation comparative pour la requête 5.

Le système ne propose pas la recette 1439 car il est nécessaire de savoir que le pancake et le waffle ont la même pâte. Il faudrait disposer d'une ressource sur la nature des préparations de base. Ensuite si les équivalences pomme - poire sont retrouvées. Le système ne propose pas de modifier « berries » par pear mais cela nous semble envisageable et créatif. **Dans ce cas il ne s'agit pas d'erreur du système mais d'un manque de connaissances.**

Liste des sources externes utilisées pour l'évaluation

– Cooking Dictionary - Ingredients Substitutions and Equivalents accessible via `http://www.gourmetsleuth.com/equivalents_substitutions.asp`?

- Ressources pour équivalence entre aliments
  - http://www.wisegeek.com
  - http://www.bitesizecooking.com/
  - http://www.astray.com/
  - http://www.whfoods.com/
  - http://arthritis.about.com/cs/gout/a/foodstoavoid.htm
  - http://www.cdkitchen.com/features/glossary/definition/
  - http://www.bigoven.com/
  - http://www.wikihow.com/
- http://cookingtips.cookingcache.com/general-cooking-tips.html
- http://www.alacuisine.net/trucastuces_cuisine.htm

Une approche en spirale pour le raisonnement à partir de cas Application : le diagnostic médical dans les services d'urgence

Bilal HUSSEIN [1], Aref MHANNA [2], Yahia RABIH [3]
[1] Modern University for Business and Science – husseinbilal@yahoo.fr
[2] Université Libanaise – aa.mhanna@hotmail.com
[3] Université Libanaise – yrabih@hotmail.com

Résumé

Dans le domaine médical, l'utilisation de l'approche de raisonnement à partir de cas (RàPC) semble particulièrement intéressante dans le sens où le type de raisonnement utilisé dans la démarche diagnostique repose essentiellement sur l'expérience liée à la résolution de cas rencontrés dans la pratique. La présentation des cas du passé au médecin permet en effet à celui-ci de s'orienter rapidement vers une solution et permet de justifier ces solutions par l'exemple. Le RàPC est ainsi une alternative possible à la problématique de la gestion des connaissances dans le domaine médical.

Plusieurs travaux concernant le RàPC en médecine ont été réalisés dans le monde et en particulier en France. Mais peu se sont occupés de services d'urgence. Donc, notre article s'intéresse à l'introduction d'une approche en spirale pour améliorer l'utilisation d'un système RàPC dans les services d'urgence en y montrant le principe d'adaptation de toutes les phases d'un système RàPC aux étapes de diagnostic et la prise de décision dans les services d'urgence. L'article présente aussi l'approche en spirale pour construire un cas source en s'appuyant sur le principe des cas prototypes et leur cycle de vie.

Mots-clés: Raisonnement à partir de cas, raisonnement en spirale, prototype, diagnostic médical, décision médicale, services d'urgence.

1 Introduction

Une étude des principales méthodes d'aide à la décision utilisées dans le domaine médical (Système Expert, Réseaux de Neurones, Raisonnement à Partir de Cas…) a montré que le Raisonnement à Partir de Cas (RàPC) [1][2][3] semble être l'approche la plus apte au raisonnement médical qui repose essentiellement sur l'expérience liée à la résolution de cas rencontrés dans la pratique [4]. Contrairement à un système expert [5], le RàPC ne nécessite pas de construire une base de connaissances et un moteur d'inférence pour la représentation des connaissances médicales. De même l'évolution d'un RàPC n'a pas besoin de l'intervention du développeur ni du programmeur, il suffit qu'il soit géré et manipulé par des acteurs médecins. De plus, l'approche RàPC est a priori bien adaptée pour fournir une justification en proposant les cas résolus du passé. Le RàPC a une capacité d'apprentissage par acquisition de nouveaux cas lui permettant ainsi d'être évolutif.

Dans le domaine médical, l'utilisation de l'approche RàPC [6] semble particulièrement intéressante dans le sens où le type de raisonnement utilisé dans la démarche diagnostique repose essentiellement sur l'expérience liée à la résolution de cas rencontrés dans la pratique. La présentation des cas du passé au médecin permet en effet à celui-ci de s'orienter rapidement vers une solution et permet de justifier ces solutions par l'exemple. Le RàPC est ainsi une alternative possible à la problématique de la gestion des connaissances dans le domaine médical.

Plusieurs travaux concernant le RàPC en médecine ont été réalisés dans le monde et en particulier en France. Parmi ces travaux nous présentons quelques systèmes: CASEY [7] est dédié au diagnostic cardiaque, PROTOS [8] pour les problèmes d'audition, ISIS [9] est dédié à la radiologie, KASIMIR non protocolaire [10] est dédié au traitement du cancer du sein, MNAOMIA [11] est dédié aux troubles du comportement alimentaire et enfin IDEM [12] concerne l'imagerie médicale.

Ainsi les travaux de recherche des systèmes RàPC dans les services d'urgence sont peu nombreux par rapport aux autres domaines médicaux. Le diagnostic et la décision d'action dans le service d'urgence doit être réalisé plus rapidement que dans d'autres services, à cause de contraintes matérielles comme le nombre limité de lits, mais aussi à cause de la gravité présumée et de la possible évolution de la pathologie [13].

Notre article s'intéresse à l'implantation d'un système RàPC dans les services d'urgence en y montrant le principe d'intégration de toutes les phases d'un système RàPC aux étapes de diagnostic et la prise de décision dans les services d'urgence. L'article présente aussi une approche en spirale pour la construction des cas sources en s'appuyant sur le principe des cas prototypes et leur cycle de vie. Cette approche consiste aussi à démarrer le raisonnement en spirale avec un cas cible possédant le minimum de descripteurs (premiers descripteurs identifiés par le médecin au cours d'un examen clinique). En passant d'une spirale à une autre le cas cible initial est enrichi par d'autres descripteurs (progression du diagnostic) et la remémoration se fait sur la liste des cas sources renvoyés par la première spirale. Ceci nous permet de manipuler moins de cas sources dans chaque spirale et d'éviter le temps de calcul long.

2 Le Raisonnement à Partir de Cas (RàPC)

2.1 Principe

Le RàPC, plus connu sous le nom de CBR (Case Based Reasoning), est un paradigme de résolution de problèmes fondé sur la réutilisation d'expériences passées pour résoudre de nouveaux problèmes [14] appelés cas cibles. Le RàPC s'appuie sur cinq opérations de base ; l'élaboration, la remémoration, l'adaptation, la révision et l'apprentissage gravitant autour d'une base de connaissances (appelée base de cas) du domaine d'application. Un cas représente notamment un problème et la solution qui lui a été appliquée.

La base de cas est une collection de cas de résolution de problèmes appelés cas sources. Elle tient une place très importante dans la mise en place d'un RàPC de qualité. Le problème de la représentation des cas commence toujours par le choix d'informations que l'on veut stocker, trouver la structure correspondante, organisation des cas et l'indexation [1][2]. Dans notre approche, à chaque cas médical est associée une collection de Prototype d'un Cas Médical (PCM) qui évolue dans un contexte médical très varié. Nous avons choisi comme application pour notre approche les services d'urgences dont l'évolution du PCM est rapide. Les services d'urgences fournissent un support d'information trop puissant pour la base de cas.

2.2 Application à la médecine d'urgence

La problématique prise en compte dans cet article est l'identification rapide d'un cas médical dans le service d'urgence. L'identification consiste à préciser le plan d'action médical à appliquer sur le malade dans l'urgence. Dans la médecine d'urgence l'identification d'un cas médical est un processus complexe imposé par trop de contraintes et entraîne dans certaines situations ce qu'on appelle « l'erreur de diagnostic ».

La médecine d'urgence se caractérise par une pression temporelle souvent forte, variable au cours de la journée et de la semaine. La fréquentation est en rapide augmentation depuis dix ans, à moyens matériels et personnels quasi constants. Les difficultés sont accrues par le caractère dynamique de la situation : le diagnostic doit être réalisé plus rapidement que dans d'autres services, à cause de contraintes matérielles comme le nombre limité de lits, mais aussi à cause de la gravité présumée et de la possible évolution de la pathologie [13].

Les erreurs de diagnostic dans les services d'urgence sont plus importantes que celles dans des autres contextes médicaux (cliniques, hôpitaux, etc.). Cet article propose une approche de raisonnement médical dans les services d'urgence à l'aide d'un système de raisonnement à partir de cas (RàPC) pour réduire au maximum les erreurs de diagnostic.

Le cycle de raisonnement médical (CRM) avec un RàPC dans les services d'urgence versus le cycle classique d'un système RàPC (table 1):

Cycle de raisonnement médical avec un RàPC	Cycle classique d'un RàPC
1- Description d'un nouveau cas (minimum de descripteurs)	Elaboration dans les RàPC classique.
2- Sélection des PCM les plus proches similaires (maximum de descripteurs égaux)	Remémoration dans les RàPC classique.
3- Adaptation du PCM choisi au cas cible ; cette étape permet de construire une solution pour le nouveau cas en se basant sur les PCM sélectionnés à l'étape précédente.	Adaptation dans les RàPC classique.
4- Révision de la solution qui permet de l'affiner grâce à son évaluation.	Révision dans les RàPC classique.
5- Mise à jour des descripteurs du cas initial et naissance d'un nouveau PCM.	Apprentissage dans les RàPC classique.

Table 1: CRM versus RàPC classique

Le processus graphique du CRM (figure 1) :

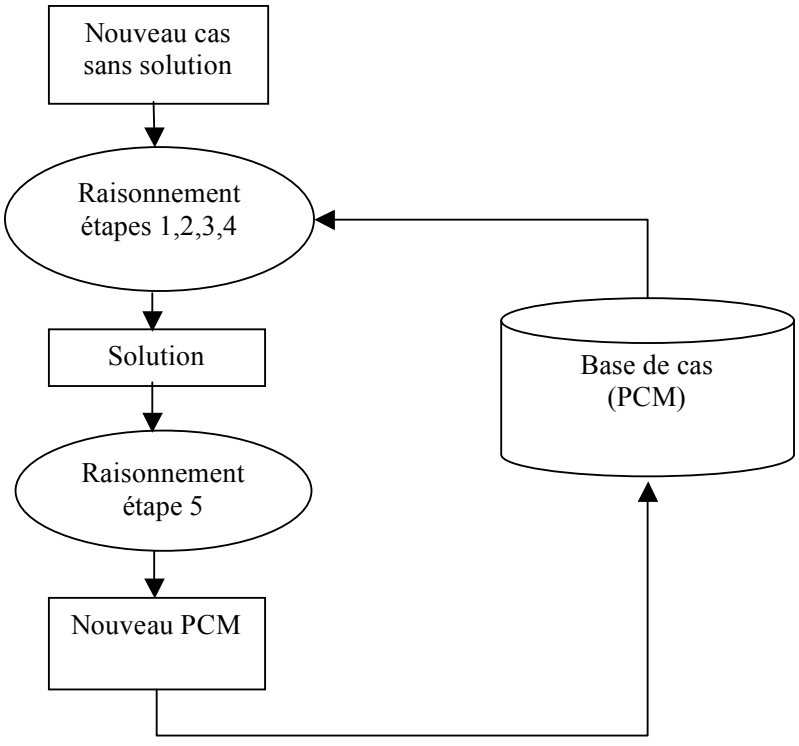


Figure 1: Processus CRM

L'erreur de diagnostic est souvent considérée comme un défaut ou une succession de défauts qui sont apparus dans le raisonnement ou le jugement du médecin (que les causes soient internes ou externes au raisonnement) et qui ont eu pour effet l'élaboration d'un diagnostic et/ou d'une décision médicale inappropriés [13]. Cependant, le CRM fournit aux médecins d'urgence un outil d'aide au raisonnement et au jugement afin de réduire autant que possible les erreurs de diagnostic.

L'efficacité d'un raisonnement à l'aide d'un CRM est proportionnelle au nombre de descripteurs évalués pour un PCM ; plus le PCM comporte de descripteurs et plus le raisonnement est efficace.

3 Représentation des cas par prototypes dans un système de RàPC médical

Les cas se présentent comme les meilleurs supports de représentation des connaissances médicales [15]. Les connaissances des médecins sont formées des connaissances générales acquises des livres médicaux, de leurs expériences sous forme de cas qu'ils ont traités eux-mêmes et de communication avec leurs collègues dans le domaine. En général, les cas n'ont pas la même importance. Certains sont typiques, alors que d'autres sont plutôt exceptionnels, par exemple, un pédiatre ne se souvient pas de tous ses patients avec la rougeole, mais peut-être ceux qui avaient des complications graves ou de ceux où leur diagnostic de rougeole a été étonnamment faux.

Les systèmes médicaux de raisonnement à partir de cas (RàPC) doivent prendre en compte le raisonnement des médecins. Ces systèmes ne devraient pas seulement être construits des connaissances du domaine général de la médecine et d'une base de cas à plat, mais la base de cas devrait être structuré en des généralisations de cas typiques appelés prototypes. Les prototypes contiennent les caractéristiques d'un ensemble de cas [15].

Le prototype est construit à partir des dossiers médicaux des patients possédant la même maladie. Degoulet [16] définit le dossier médical comme suit : « Le dossier du malade ne se résume pas à l'observation écrite du médecin (le dossier médical proprement dit) ou aux notes de l'infirmière (le dossier infirmier). Il englobe tout ce qui peut être mémorisé chez un malade, des données démographiques aux enregistrements électro-physiologiques ou aux images les plus sophistiquées. Compte tenu de ce rôle, le dossier du malade est et restera longtemps l'outil principal de centralisation et de coordination de l'activité médicale. »

Le dossier médical possède des caractéristiques, parmi lesquelles on peut citer les éléments suivants [17]:

– il concerne tout d'abord des individus ;

– il contient des documents papier, des radiographies, des images échographies, etc.

– il contient aussi des données stables (durée de vie très longue) et des données dont la durée de vie est limitée. Par exemple, le rythme cardiaque est relevé de façon très fréquente en réanimation, mais une fois cette réanimation terminée, les valeurs relevées ne sont généralement plus utiles. A l'opposé, la date de naissance est une donnée médicale très stable et qui a une durée de vie très longue.

– le dossier médical est un outil de collaboration et de coordination entre différents types d'acteurs.

Cependant, le Prototype d'un Cas Médical (PCM) sera fondé sur des informations extraites de dossiers médicaux. Nous appelons prototypage le cycle de vie permettant de construire un prototype à partir de dossiers médicaux.

Le prototypage est le cycle de vie d'un PCM décrivant toutes les étapes par lesquelles passe le PCM, depuis sa naissance jusqu'à son utilisation et évolution dans des situations réelles. Le prototypage d'un PCM est formé des étapes suivantes : analyse des dossiers médicaux, conception, création du PCM et utilisation et évolution (Figure 2).

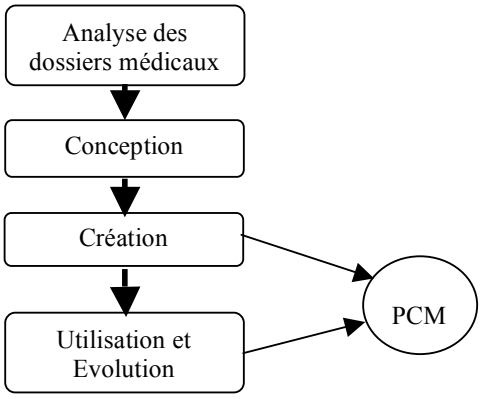


Figure 2: Cycle de vie d'un PCM

Etape 1 : Analyse des dossiers médicaux

Le dossier médical comporte l'histoire médicale d'un patient ainsi que les soins qui lui sont prodigués. Le dossier médical est le support d'informations et de communication entre les acteurs du domaine médical. En raison de la complexité du dossier médical son analyse devient une opération difficile. Le dossier médical est analysé de façons différentes et en conséquence les résultats d'analyse ne sont pas les mêmes. De ce point de vue et après une analyse de dossiers médicaux, il sera nécessaire de définir un cas médical avec le maximum d'informations descriptives.

Afin de faciliter l'analyse de dossiers médicaux nous proposons d'utiliser des dossiers semi-structurés lors de la saisie d'informations. Les documents semi-structurés permettent une saisie plus naturelle et un échange plus simple de l'information [18].

Etape 2 : Conception

Un diagnostic médical est basé sur le dossier médical. Le diagnostic est dit complet si le dossier médical comporte toutes les informations requises pour le médecin. Notons ici que ces informations relèvent de l'examen clinique, de l'examen biologique et de l'imagerie médicale.

Quatre ensembles de descripteurs décrivent un cas médical. L'examen Clinique fournit un ensemble initial de descripteurs appelé (***CL***) contenant les données stables, les données historiques et les symptômes du patient. Les résultats de l'examen biologique et de l'imagerie médicale forment deux ensembles d'extension de descripteurs appelés respectivement (***BI***) et (***IM***) (Figure 3).

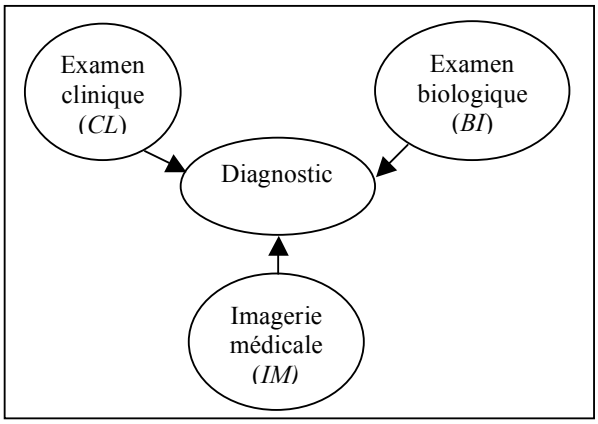


Figure 3: Diagnostic médical

- L'ensemble ***CL*** est formé de trois sous-ensembles: données stables, données historiques et les symptômes

Sous-ensemble données stables : La durée de vie des données stables est longue, par exemple la date de naissance, l'origine, etc.

Sous-ensemble données historiques : elles peuvent être stables ou non, par exemple une maladie de naissance est une donnée stable alors qu'une température élevée due à la présence d'un virus est une donnée non stable.

Sous-ensemble données symptômes : elles sont les données prélevées au moment de l'examen clinique, par exemple température, pression, battement de cœur, respiration rapide, etc.

- L'ensemble ***BI*** est formé de plusieurs sous-ensembles, par exemple l'hémogramme, le bilan lipidique, le bilan glycémique, le bilan de la fonction rénale, le bilan hépatique, etc. Chaque sous-ensemble est associé à un type d'analyse et comprend plusieurs paramètres par exemple :

Sous-ensemble hémogramme : cet ensemble contient les données permettant d'explorer l'anémie, les défenses immunitaires, les plaquettes et les maladies du sang. Par exemple, hématie, hémoglobine, hématocrite, leucocyte, etc.

Sous-ensemble bilan lipidique : cet ensemble contient les données permettant d'explorer le risque cardiovasculaire. Par exemple, triglycérides, cholestérol, HDL, LDL, etc.

- L'ensemble ***IM*** est formé de plusieurs sous-ensembles dépendants de la technique utilisée au moment de l'examen d'imagerie médicale. Parmi ces techniques nous citons les rayons X, les IRM, les échographiques, etc. Chaque technique définit un sous ensemble de données extraites de l'analyse des radios faite par un radiologue.

Les ensembles CL, BI et IM qui caractérisent un dossier médical forment la partie problème du cas médical. Le but final du diagnostic médical est d'identifier la maladie et proposer le traitement convenable. Donc, le diagnostic fournit le couple <maladie, traitement>. On note ici, que ce couple est la solution du problème posé par le cas médical.

Etape 3 : Création d'un PCM

Le cas médical est une entité à deux dimensions le problème et la solution. Le problème est caractérisé par les trois ensembles de l'analyse du dossier médical. La solution est caractérisée par le couple <maladie, traitement>.

La création initiale de l'entité « Cas médical » est le prototype PCM. Ce prototype est créé par des médecins experts avec le minimum de paramètres possibles. Ces paramètres descripteurs PCM sont évalués et utilisés dans l'étape d'utilisation et évolution.

Etape 4 : Utilisation et évolution

Le PCM est utilisé dans des contextes très variés et complexes et ceci pour plusieurs raisons, la plus importante est la multiplication des acteurs (patient, infirmiers, médecins, parents, employés, etc.). L'utilisation du PCM dans un contexte réel a des conséquences positives. Le PCM peut évoluer (augmentation du nombre de descripteurs) et il peut faire naître un nouveau PCM c'est un phénomène de prototypage en spirale (Figure 4.).

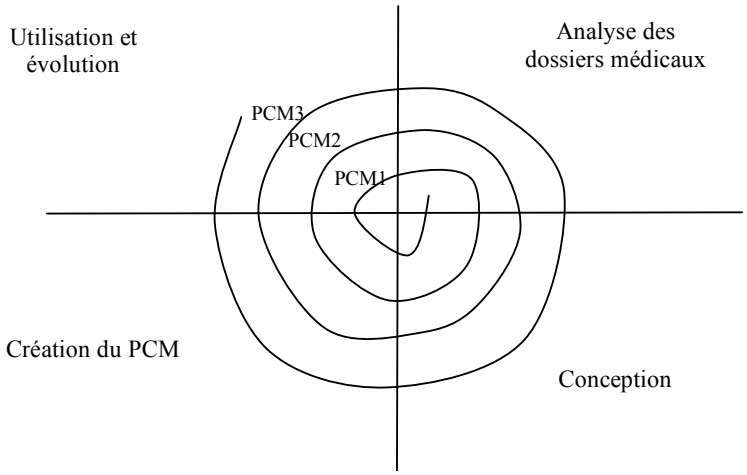


Figure 4: Spirale d'évolution du PCM

5 Un raisonnement en spirale

Le raisonnement CRM est basé sur un nombre de descripteurs caractéristiques d'un PCM. Dans l'étape de conception d'un PCM les descripteurs sont retirés depuis les dossiers médicaux et ont été classés par des ensembles et des sous-ensembles.

La phase de sélection des PCM, qui consiste à retrouver dans la base de cas des cas similaires, est en partie basée sur une mesure de similarité. En général, on passe tout d'abord par une recherche d'un ensemble de cas compatibles à l'aide d'une indexation définie au préalable puis la mesure de similarité va permettre de sélectionner les cas similaires parmi ce sous ensemble. Nous n'avons pas abordé le problème de l'indexation dans nos travaux pour le moment.

Le raisonnement médical classique est itératif c'est-à-dire il part d'un ensemble de descripteurs minimal et il progresse dans le temps avec des nouveaux descripteurs. Cependant, le CRM réagit de la même façon mais en spirale. A partir d'un cas cible on raisonne sur un ensemble de PCM sélectionnés jusqu'à la naissance d'un nouveau PCM (un nouveau cas cible). Dans la spirale suivante le raisonnement se fait sur l'ensemble des PCM sélectionnés dans la précédente, seuls les nouveaux descripteurs sont considérés dans la mesure de la similarité. A la fin de cette spirale, on aura la naissance d'un nouveau PCM et un ensemble de

PCM plus petit. L'itération en spirale s'arrête avec la réussite d'une solution adaptée. Plusieurs avantages sont derrière cette technique de sélection :

- le cas cible est construit une seule fois même en présence de manque de descripteurs.
- l'ensemble de cas sources diminue avec chaque spirale
- l'acteur (médecin) n'est pas en face d'une grande liste de cas similaires
- le nombre de descripteurs augmente en fonction du nombre de spirales demandées

5 Conclusion

L'article présente une méthode de remémoration itérative dite "en spirale" qui permet d'impliquer le médecin dans l'étape de remémoration des cas sources. L'approche est appliquée dans le domaine du diagnostic médical dans les services d'urgence dans le but d'éviter les erreurs de diagnostic et de fournir un support à la décision par la présentation d'expériences concrètes. Le RàPC est a priori le système le plus capable de suivre le raisonnement médical dans les services d'urgence. Cet article propose une approche pour améliorer l'utilisation du système RàPC dans ce domaine. L'approche de raisonnement en spirale est très pertinente dans les domaines qui nécessitent un raisonnement rapide sur des données incomplètes, et c'est le cas des services d'urgence où le raisonnement médical est une tâche particulièrement difficile à cause de la qualité des données et de la complexité de la tâche de prise de décision médicale. Les algorithmes de mesure de similarité, adaptés à notre approche en spirale, jouent un rôle très important pour la performance d'un système RàPC médical. Le choix de l'algorithme le plus pertinent à notre approche en spirale sera étudié dans nos prochains articles.

6 Références

[1] Aamodt A., Plaza E., « Case-based reasoning: Foundational issues, methodological variations, and system approaches ». *AICOM*, **7**, pp 39–59, 1994.

[2] Kolodner J., « Case-based reasoning ». In Morgan Kaufmann Pub., 1993.

[3] Mille A., « Etat de l'art du Raisonnement à Partir de Cas », In : Plateforme de l'AFIA 99; Palaiseau, 1999.

[4] Sefion I., Gailhardou M., Ennaji A., « Le Raisonnement à Partir de Cas pour l'aide à la prise en charge de l'asthme », Ingénierie des systèmes d'information ISSN 1633-1311, 2003, vol. 8, no 1, pp. 11-32,

[5] Fargeas X., Frydman F., « Les Systèmes Experts en Médecine », Paris : Hermes, 1988.

[6] http://www.med.uni-rostock.de/WebServerDaten/IMIB/HTML/english/research/rese arch.html

[7] Koton P., Using experience in learning and problem solving, PhD Thesis, Laboratory of Computer Science MIT, 1988.

[8] Bareiss E. R., « PROTOS : A Unified Approach to Concept Representation, Classification and Learning », PhD Thesis, University of Texas, 1988.

[9] Kahn CE Jr., Case-based selection of diagnostic imaging procedure, In: Leake DB(ed). Case-Based Reasoning: Papers from the 1993 Workshop. Menlo Park : AAAI Press, 1993.

[10] Bresson B., Lieber J., Raisonnement à partir de cas pour l'aide au traitement du cancer du sein, In : Actes des journées ingénierie des connaissances, 2000 ; pp 189-196.

[11] Bichindaritz I., Apprentissage de concepts dans une mémoire dynamique : Raisonnement à Partir de Cas adaptable à la tâche cognitive, Thèse de doctorat, Université René Descartes, 1994.

[12] Le Bozec C, Jaulent MC, Zapletal E., IDEM : Remémoration de cas pour l'aide au diagnostic en Anatomie Pathologique. In : Ingénierie des connaissances. J.Charlet, M.Zacklad, G.Kassel, D.Bourigault (eds). Paris : Eyrolles, 2000; pp 371-386.

[13]Marquié L., Raufaste E., Mariné C., Ecoiffier M., « L'erreur de diagnostic en médecine d'urgence : application de l'analyse rationnelle des situations de travail », Presses Universitaires de France, Le travail humain, 2003/4 - Volume 66, ISSN 0041-1868 | ISBN 2130536042 | pp 347-376

[14] Cordier A., Fuchs B., « Un assistant pour la conception et le développement des systèmes de Ràpc » , 13ème Atelier Ràpc, 2005.

[15] R. Schmidt, T. Waligora, and O. Vorobieva , Prototypes and Case-Based Reasoning for Medical Applications, Studies in Computational Intelligence (SCI) 73, pp 285–317 (2008)

[16] Degoulet P., Fieschi M., Traitement de l'information médicale : méthodes et applications hospitalières, Masson, Paris, 1991.

[17] Laforest F., Frénot S., Al Masri N. Dossier médical semi structuré pour des interfaces de saisie multimodales, Lavoisier, Document numérique, Volume 6, ISSN 1279-5127, p. 29- 46, 2002/1-2

[18] Bryan M. , (Ed) Guidelines for using XML for Electronic Data Interchange, 1998, http://www.xmledi-group.org/xmledigroup/guide.htm