Le moteur de raisonnement à partir de cas de WikiTaaable

Jean Lieber

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 [21]), 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* [22]) : 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 $\texttt{pb} \overset{op}{\longmapsto} \texttt{pb}'$ 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 [19]. On aurait également pu choisir le terme « opérateur d'adaptation » qui est utilisé dans [7].

Ce qui conduit à une adaptation appliquant successivement $\mathcal{A}_{\underrightarrow{\sigma_1}}, \ldots, \mathcal{A}_{\underrightarrow{\sigma_p}}, \mathcal{A}_{\underleftarrow{\gamma_q}}, \ldots$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. Dans l'application WIKITAAABLE, un cas représente une recette et son découpage en une partie problème et une partie solution est arbitraire. On utilise plutôt, à la place de la notion de problème, la notion de requête, sachant qu'une requête peut s'interpréter comme l'ensemble des recettes qu'elle satisfait. On peut ainsi assimiler la requête Q donnée par l'utilisateur à un problème cible : $Q = \texttt{cible}$. De quels problèmes une recette $\texttt{Sol}(\texttt{srce}) = R$ est-elle la solution ? Puisqu'un problème est une requête, l'ensemble des requêtes `srce` auxquelles R répond sont des candidats. En particulier, si on suppose que le langage des requêtes contient le langage des recettes[3], alors R est un problème dont R est une solution[4]. Avec ce choix, on aboutit à $\texttt{srce} = R$ et donc, le cas source sera le couple $(\texttt{srce}, \texttt{Sol}(\texttt{srce})) = (R, R)$. On comprendra qu'avec ce choix, la distinction problème-solution est très artificielle pour WIKITAAABLE. Elle ne sera plus faite dans le reste de l'article (pas plus que la notion de reformulation, d'ailleurs).

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 » (*knowledge containers* [20]) :

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

`KB` 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`) 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 médierranéennes. 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.

[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.

`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 [5]. Une recette $R \in$ `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[6] (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[7] 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 $idx(R) \nvDash_{\mathcal{O}} a$ alors le terme « *Rien d'autre* » de (1) est une conjonction de littéraux contenant le littéral $\neg a$.[8] Par exemple, cette hypothèse du monde clos permet de déduire que $idx(R) \vDash_{\mathcal{O}} \neg\texttt{viande} \wedge \neg\texttt{poisson}$, i.e., que R est une recette végétarienne[9].

On peut noter que $idx(R)$ a exactement un modèle. 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 $\vDash_{\mathcal{O}}$: pour $C, D \in \mathcal{H}_{idx}$, $C \vDash_{\mathcal{O}} D$ 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 $\vDash_{\mathcal{O}}$ 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 $\sigma \in$ `CA` peut être considérée comme une règle d'inférence spécifique $\frac{R \text{ est une bonne recette}}{\sigma(R) \text{ est une bonne recette}}$([10]). Une substitution est donnée par deux conjonctions de 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

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

[6] 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.

[7] 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, 18]).

[8] Si f et g sont deux formules propositionnelles alors $f \vDash_{\mathcal{O}} g$ 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.

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

[10] 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[11].

[11] 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[12].

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

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 $(\sigma_2 \circ \sigma_1)^{-1} = \sigma_1^{-1} \circ \sigma_2^{-1}$.

coût. L'inférence effectuée par le moteur de RÀPC s'appuie sur des substitutions, qui sont soit issues de 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 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

$$idx(R) \overset{\sigma_1}{\longmapsto} \ldots \overset{\sigma_p}{\longmapsto} \Sigma(idx(R)) \models_{\mathcal{O}} \Gamma(Q) \overset{\gamma_q}{\longmapsto} \ldots \overset{\gamma_1}{\longmapsto} Q \tag{3}$$

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 $\models_{\mathcal{O}}$ et la spécialisation aux substitutions γ_q^{-1}, …, γ_1^{-1}). Pour résumer, la recette R suit la chaîne de transformations suivante :

$$R \overset{\sigma_1}{\longmapsto} \ldots \overset{\sigma_p}{\longmapsto} \Sigma(R) \overset{\gamma_q^{-1}}{\longmapsto} \ldots \overset{\gamma_1^{-1}}{\longmapsto} \Gamma^{-1} \circ \Sigma(R) \tag{4}$$

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

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

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) :

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

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 laitue par scarole (cf. $\models_{\mathcal{O}}$ 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 :

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

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 généralier `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[13].

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[14] 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[15]. On définit alors le coût d'un 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., $idx(R) \vDash_{\mathcal{O}} Q$), alors ce coût est minimal et fixé à 0. Par ailleurs, on fait l'hyothèse que cette mesure est additive, autrement dit que le coût du chemin donné par (7) est la somme

[13] 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 $\sum_i \texttt{coût}(\sigma_i) + \sum_j \texttt{coût}(\gamma_j) \leqslant \sum_k \texttt{coût}(g_k)$.

[14] Ce qui se ramène à la question de la signification de ce qu'est une « bonne recette » : voir note 11.

[15] 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 $R'' = (\texttt{banane} \rightsquigarrow \texttt{pomme})(R')$).

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

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

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 $a \vDash_{\mathcal{O}} b$ et $b \vDash_{\mathcal{O}} c$. 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 $a \rightsquigarrow b$ ($a \vDash_{\mathcal{O}} b$) définie ci-dessous vérifie la propriété (9) :

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

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

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

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[16]. 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 $\texttt{margarine} \rightsquigarrow \texttt{beurre}$).

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

[16] 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.

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.

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[17]) 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 $\mathit{idx}(R_{\texttt{tap}}) \models_{\mathcal{O}} g(Q) \longleftarrow\!\shortmid Q$ 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 »).

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 correspond à une classe de recettes. 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...

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

La solution actuellement envisagée est la suivante. On définit comme avant $idx(R)$ 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. $idx(R)$ 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 $idx(R) \not\models_{\mathcal{O}} b$ (par exemple, $b =$ `viande`, mais aussi $b =$ `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.[18]

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 `jus_de_citron` $\Rightarrow$ `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 envisagent à passer à un formalisme utilisant des attributs[19]. 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 ?[20] Pas nécessairement : il y a du lait en poudre ($(\mathtt{lait\_en\_poudre} \Rightarrow \mathtt{lait}) \in \mathcal{O}$) et du beurre allégé ($(\mathtt{beurre\_allégé} \Rightarrow \mathtt{beurre}) \in \mathcal{O}$). Par conséquent, $\mathcal{O}$ doit respecter le fait que `lait` $\not\models_{\mathcal{O}}$ `liquide` et `beurre` $\not\models_{\mathcal{O}}$ `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`)[21]. Comment cela s'articule avec les inférences du système est un (ensemble de) problème(s) ouvert(s)...

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.

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

[19] 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.

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

[21] 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`.

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.

Remerciements

L'auteur tient à remercier les personnes impliquées dans le projet TAAABLE pour le plaisir qu'il a à travailler avec elles.

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 : Soumis au 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.). – Représentation du temps dans des recettes de cuisine, remémoration et adaptation — Utilisation d'une algèbre temporelle. *In : Soumis au 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] 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.

[19] 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.

[20] 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.

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

[22] 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.