

Construction manuelle de la partie haute d'une ontologie modulaire destinée à une annotation de cas textuels — étude de cas pour une application culinaire dans le cadre du projet TAAABLE

Pierre-Antoine Champin¹, Amélie Cordier¹, Sylvie Després²,
Béatrice Fuchs¹, Jean Lieber³, Alain Mille¹

¹ LIRIS (UMR 5205, CNRS, Université de Lyon), prenom.nom@liris.cnrs.fr

² LIPN (UMR 7030, CNRS, Université Paris 13), sylvie.despres@lipn.univ-paris13.fr

³ Équipe Orpailleur, LORIA (UMR 7503 CNRS, INRIA, Université de Nancy), lieber@loria.fr

Résumé

Un système de raisonnement à partir de cas utilise plusieurs bases de connaissances, en particulier les cas, les connaissances d'adaptation et les connaissances du domaine. Ces dernières se présentent souvent sous la forme de classes (ou concepts) utiles pour représenter et organiser les cas, et de liens entre ces classes (liens de subsumption, de composition, etc.) ; elles peuvent donc prendre la forme d'une ontologie. Le système de RÀPC TAAABLE¹ est en cours de conception. Il a pour objectif la résolution de problèmes culinaires et s'appuie sur l'ontologie POTAAABLE² qui est en cours de développement et qui utilisera le langage OWL DL. Cet article présente le travail de conception de la partie de POTAAABLE contenant les concepts culinaires généraux (recettes, aliments, actions culinaires, etc.) et leurs liens. Bien que cette ontologie se veuille partageable et réutilisable pour d'autres applications dans le domaine culinaire, les priorités dans son développement sont dictées par les besoins du système TAAABLE. En particulier, l'adaptation devrait essentiellement se faire par substitutions de types d'aliments, et donc le concept de type d'aliment et les concepts « connexes » ont une priorité importante lors du développement de l'ontologie.

Mots clés : ontologie de domaine, cas textuels, application culinaire.

1 Introduction

Dans ce papier, nous relatons un travail qui rassemble, dans le cadre du projet TAAABLE, les compétences multiples de plusieurs membres de la communauté RÀPC nationale. Le projet TAAABLE réunit une vingtaine de chercheurs et enseignants-chercheurs du LIPN (CNRS, Paris13), du LIRIS (CNRS, Lyon1) et du LORIA (CNRS, INRIA, Université de Nancy). Il constitue par conséquent l'occasion de rapprocher des chercheurs de domaines différents et a pour ambition de les faire collaborer. Cette collaboration est la suite logique des travaux communs engagés depuis quelques années par le LIRIS et le LORIA sur l'apprentissage incrémental de connaissances. Des compétences dans le domaine du traitement automatique de la langue, de la fouille de textes et de l'ingénierie des ontologies à partir de textes sont maintenant représentées.

Le contexte du projet est celui de la compétition CCC, *Computer Cooking Contest*³ organisée dans le cadre de la conférence ECCBR dont la première session aura lieu le 1^{er} septembre 2008. La participation à cette compétition permettra également d'augmenter notre visibilité au plan international et nous donnera l'occasion de nous comparer à d'autres équipes.

¹TAAABLE n'est pas un acronyme, c'est un nom, le nom du projet TAAABLE.

²Aussi surprenant que cela puisse paraître, POTAAABLE est un acronyme. C'est même un acronyme récursif. POTAAABLE signifie « POTAAABLE est l'Ontologie de TAAABLE ».

³Computer Cooking Contest, <http://www.wi2.uni.trier.de/eccbr08/index.php?task=ccc>

Le système à construire doit proposer une recette répondant au mieux à une requête exprimée en texte libre. Plusieurs tâches sont à réaliser : une tâche obligatoire qui est de répondre à une requête impliquant la sélection et la modification d'une recette pour un plat unique ; une tâche dite de négation où le système doit proposer une recette sachant que certains ingrédients sont à éviter et qu'ils sont dénotés par un terme ou comme appartenant à une catégorie d'ingrédients ; enfin une tâche de création de menu est également proposée. Seuls les ingrédients figurant les recettes de la base peuvent être utilisés.

Le choix que nous avons fait est la construction d'un système de RÀPC dont les cas seront construits à partir de textes décrivant des recettes de cuisine. Les recettes seront annotées en utilisant l'ontologie culinaire POTAAABLE construite dans le cadre de ce projet. Les tâches de remémoration et d'adaptation du système pour produire des réponses satisfaisantes aux requêtes dépendent par conséquent de la qualité de la construction des cas à partir des recettes. L'élaboration des cas repose également sur l'utilisation de l'ontologie du domaine de la cuisine.

Si nous adoptons la définition de l'ontologie proposée par Gruber [6], « Une ontologie est une spécification formelle d'une conceptualisation d'un domaine, partagée par un groupe de personnes, qui est établie selon un point de vue imposé par l'application construite », alors un tel modèle doit faciliter la description et l'organisation des éléments de connaissances du système (cas et connaissances d'adaptation) et permettre une annotation sémantique des textes des recettes qui sera utile pour la peupler.

L'ontologie construite sera constituée d'un ensemble de concepts à la fois organisés hiérarchiquement et structurés par les relations liant ces concepts et comportera des règles et des axiomes. Le langage de représentation adopté est OWL. Les choix méthodologiques sont la construction d'une ontologie modulaire et la réutilisation de ressources ontologiques jugées pertinentes pour le système à construire. La construction est décomposée en deux étapes : (1) la construction manuelle de la partie de l'ontologie constitués des modules généraux et propriétés les liant, relatifs aux connaissances du domaine (2) la construction semi-automatique de l'ontologie propre à chacun des modules et (3) le peuplement de cette ontologie grâce à des techniques semi-automatiques de traitement automatique du langage naturel.

Cet article traite du point (1) et discute de son articulation avec les points (2) et (3). Pour simplifier la présentation, nous appelons partie haute de l'ontologie le modèle des modules généraux et les propriétés qui les lient et la partie basse celle qui correspond au modèle associé aux différents modules.

Après avoir rappelé quelques notions sur le RÀPC et les ontologies (section 2), le projet TAAABLE est présenté (section 3). La section 4 constitue le cœur de cet article : elle présente le haut d'ontologie qui a été formalisé, avec des principes sur sa construction et un exemple d'annotation de cas textuel à l'aide de cette ontologie. L'articulation avec le travail sur la spécialisation de l'ontologie est discutée à la section 5. La section 6 conclut cet article et décrit les perspectives du projet TAAABLE.

2 Rappels : le RÀPC et les ontologies

2.1 Le RÀPC

On se place dans un domaine d'application particulier pour lequel les notions de problème et de solution sont bien définies. Si pb est un problème (resp., sol est une solution) alors pb (resp., sol) représente un problème (resp., une solution) de ce domaine. Par ailleurs, on suppose qu'il existe une relation binaire liant un problème à une solution dont la signification est « a pour solution ». Dans certaines applications du RÀPC, cette relation n'est qu'imparfaitement connue. En revanche, on dispose d'une base de cas (appelés cas sources), c'est-à-dire d'un ensemble de couples $srce-case = (srce, Sol(srce))$ où $srce$ est un problème, $Sol(srce)$ est une solution et $srce$ a pour solution $Sol(srce)$. On note CD la base de connaissances codant les connaissances du domaine.

Raisonnement à partir de cas, c'est résoudre un problème, appelé problème cible et dénoté par tgt , au moyen de la base de cas. Ce raisonnement est d'habitude constitué de deux grandes étapes : la remémoration, qui a pour objectif de sélectionner un cas source jugé similaire au problème cible, et l'adaptation, qui a pour objectif de résoudre le problème cible en s'appuyant sur le cas source remémoré.

Le RAPC textuel est une branche du domaine du RAPC pour laquelle les cas sont des textes (en langue naturelle⁴) ou des documents avec des parties textuelles.

2.2 OWL-DL : langage de representation des ontologies

Langage de representation. Le langage OWL-DL est une recommandation du W3C pour representer des ontologies qui est equivalent, dans sa version actuelle, a la logique de description *SHOIN(D)* [2]. Nous ne presenterons ici que les elements de ce langage necessaires pour les besoins de cet article.

On se place dans un univers particulier dans lequel les elements sont appeles des objets. On distingue les notions de *classes* (qui representent des ensembles d’objets), de *proprietes* (representant des relations binaires entre objets) et d’*instances* (representant des objets). Parmi les proprietes, on distingue les proprietes d’objets qui etablissent des liens entre objets, et les proprietes de type de donnees qui etablissent des liens entre objets et valeurs, une valeur etant un element d’un des types de donnees predefinis de XML Schema (**float**, **string**, etc.).

Une interpretation \mathcal{I} est un couple $(\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ ou $\Delta_{\mathcal{I}}$ est un ensemble non vide et ou $\cdot^{\mathcal{I}}$ est une application qui a une classe C , une propriete p d’objets, une propriete d du type de donnee $\tau \in \{\mathbf{float}, \mathbf{string}, \dots\}$ et une instance a , associe respectivement $C^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}}$, $p^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$, $d^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}} \times \tau$ et $a^{\mathcal{I}} \in \Delta_{\mathcal{I}}$.

Les classes sont soit atomiques (i.e. des noms de classe) soit des expressions s’appuyant sur des constructeurs de classe. Le tableau 1 donne une liste des constructeurs utiles pour POTAAABLE et leur semantique. Les proprietes et les instances sont toujours atomiques.

Constructeurs de classe	Syntaxe	Semantique
Classes predefinies	\top	$\Delta_{\mathcal{I}}$
	\perp	\emptyset
Operateurs ensemblistes	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
	$\neg C$	$\Delta_{\mathcal{I}} \setminus C^{\mathcal{I}}$
Quantificateurs	$\exists p.E$	$\{x \mid \exists (x, y) \in p^{\mathcal{I}}, y \in E^{\mathcal{I}}\}$
	$\forall p.E$	$\{x \mid \forall (x, y) \in p^{\mathcal{I}}, y \in E^{\mathcal{I}}\}$
Restrictions de cardinalite	$(\leq n p)$	$\{x \mid \text{card}\{y \mid (x, y) \in p^{\mathcal{I}}\} \leq n\}$
	$(\geq n p)$	$\{x \mid \text{card}\{y \mid (x, y) \in p^{\mathcal{I}}\} \geq n\}$
	$(= n p)$	$\{x \mid \text{card}\{y \mid (x, y) \in p^{\mathcal{I}}\} = n\}$

TAB. 1 – Constructeurs de classe OWL et leur semantique. C et D denotent des classes ; p denote une propriete (d’objet ou de type de donnee) ; E denote une classe ou un type de donnee ; n denote un entier naturel ; $\text{card } e$ denote la cardinalite de l’ensemble e .

Les formules d’OWL utilisees dans cet article sont decrites dans le tableau 2, avec leur syntaxe et leur semantique. Les formules appartenant a une ontologie (dans cet article, l’ontologie culinaire) sont appelees axiomes.

Soit \mathcal{O} , une ontologie. Si f est une formule, on note $\models_{\mathcal{O}} f$ si, quel que soit le modele \mathcal{I} de \mathcal{O} , \mathcal{I} est un modele de f . Le test d’instanciation est l’inference qui consiste, etant donnee une ontologie \mathcal{O} et une classe C , a trouver toutes les instances i de \mathcal{O} telles que $\models_{\mathcal{O}} C(i)$.

⁴Nous considererons dans cet article que l’anglais est une langue naturelle, meme si elle ne l’est pas pour certains des auteurs de cet article.

	Syntaxe	Sémantique
Classes / Propriétés	$X \sqsubseteq Y$	$X^{\mathcal{I}} \sqsubseteq Y^{\mathcal{I}}$
	$X \equiv Y$	$X^{\mathcal{I}} = Y^{\mathcal{I}}$
Propriétés	domaine(p, C)	$\forall (x, y) \in p^{\mathcal{I}}, x \in C^{\mathcal{I}}$
	co-domaine(p, D)	$\forall (x, y) \in p^{\mathcal{I}}, y \in D^{\mathcal{I}}$
	fonctionnelle(p)	$\forall x \in \Delta_{\mathcal{I}}, \#\{(x, y) \in p^{\mathcal{I}}\} \leq 1$
	transitive(q)	$\forall x, y, z \in \Delta_{\mathcal{I}}, (x, y) \in q^{\mathcal{I}} \wedge (y, z) \in q^{\mathcal{I}} \implies (x, z) \in q^{\mathcal{I}}$
Instances	$C(i)$	$i^{\mathcal{I}} \in C^{\mathcal{I}}$
	$q(i, j)$	$(i^{\mathcal{I}}, j^{\mathcal{I}}) \in q^{\mathcal{I}}$
	tous-différents(S)	$\forall i \neq j \in S, i^{\mathcal{I}} \neq j^{\mathcal{I}}$

TAB. 2 – Syntaxe et sémantique des formules OWL. X et Y dénotent des classes ou des propriétés (d’objet ou de type de données) ; p dénote une propriété (d’objet ou d’un type de données τ) ; C dénote une classe ; D dénote une classe ou un type de donnée ; $\text{card } e$ dénote la cardinalité de l’ensemble e ; q dénote une propriété d’objet ; i dénote une instance ; j dénote une instance ou une valeur de τ ; S dénote un ensemble d’instances.

2.3 RÀPC textuel et cas annotés

Les travaux dans le domaine du raisonnement à partir de cas textuel ont conduit à quatre questions principales : (1) comment évaluer la similarité entre des cas textuels ; (2) comment passer des textes à des représentations de cas structurés ; (3) comment adapter des cas textuels ; (4) comment engendrer automatiquement des représentations pour le RÀPC textuel [10]. Dans le domaine du RÀPC textuel, le travail de [3] cherche à évaluer l’apport de l’ontologie dans la phase de remémoration. L’utilisation d’ontologie dans l’élaboration du cas est également soulignée. Il nous semble que l’apport possible de l’ontologie se situe également dans sa capacité à permettre des raisonnements formels. C’est ce que nous testerons dans le projet TAAABLE en évaluant l’impact de l’ontologie dans le processus d’adaptation.

3 Le système TAAABLE

Un système participant au concours doit exploiter une base de recettes de cuisine (un cas = une recette) pour pouvoir répondre à des requêtes de recettes (problèmes cibles). Une des difficultés de ce concours tient au fait que les recettes sont exprimées dans un format peu structuré : ce sont des zones de textes en anglais identifiées par trois balises : une balise pour le titre, une pour chaque ingrédient et une pour la préparation.

3.1 Exemple de cas textuel

La figure 1 donne un exemple de cas textuel⁵.

3.2 Principe envisagé du système de RÀPC de TAAABLE

L’ontologie POTAAABLE est conçue pour servir au système de RÀPC TAAABLE dont la finalité principale est la remémoration et l’adaptation des cas. Par conséquent, la définition du fonctionnement futur du système est nécessaire pour orienter les choix de construction de l’ontologie. Cette section présente le principe envisagé de ce système. On pourra noter que l’élaboration du principe envisagé ne s’est pas fait avant ni après la construction de l’ontologie mais en même temps.

⁵Les auteurs déclinent toute responsabilité concernant la valeur culinaire de cette recette.

```

<RECIPE>
  <TI>Citrus Marinade and Salsa (for Chicken or Fish)</TI>
  <IN>1/4 c Lemon juice</IN>
  <IN>2 tb Dry sherry</IN>
  <IN>2 tb Olive oil or cooking oil</IN>
  <IN>1/2 ts Dried oregano, crushed</IN>
  <IN>1/4 ts Garlic salt</IN>
  <IN>1 md Tomato, peeled, seeded and chopped</IN>
  <IN>1 Green onion, sliced</IN>
  <IN>1/4 c Sliced pitted ripe olives</IN>
  <IN>1 tb Snipped parsley</IN>
  <IN>1 tb Slivered almonds</IN>
  <PR>
  For marinade, in a 1cup measure combine lemon juice, sherry, oil,
  oregano, and garlic salt. Set aside 2 tablespoons. Pour remaining
  into a plastic bag in a bowl; add chicken or fish. Close bag;
  chill 1 hour, turning once. For salsa, combine the 2 tablespoons
  marinade, tomato, green onion, olives, parsley, and
  almonds. Cover; chill till serving time. Drain chicken or fish,
  reserving marinade. Grill till chicken is tender and fish is
  flaky, turning and brushing often with the marinade. Serve with
  chilled salsa and avocado slices, if desired. Makes 4
  servings. Nutrition facts per tablespoon salsa: 13 cal.
  </PR>
</RECIPE>

```

FIG. 1 – Un cas textuel du CCC.

Connaissances. Les connaissances de TAAABLE apparaissent sous deux formes :

- Les connaissances « formelles », qui sont exprimées dans le langage OWL DL et qui sont manipulables à travers des raisonnements ;
- Les connaissances « textuelles », qui sont exprimées sous forme textuelle⁶ (avec une petite structuration XML).

Le lien entre ces deux types de connaissances est l'*annotation*, qui permet à un (extrait de) texte d'être associé à des éléments de connaissances (instances, classes).

Nous réserverons les notations du RÀPC — *srce*, *Sol(srce)*, *tgt* — aux connaissances formelles. Les connaissances du domaine de TAAABLE sont données par l'ontologie \mathcal{O} dont la construction de la partie haute est l'objet de cet article. Un cas source représente une recette de cuisine et est une instance (directe ou indirecte) de la classe des recettes : $\models_{\mathcal{O}} \text{Recipe}(\text{srce-case})$. Une requête est représentée par une sous-classe R de la classe **Recipe** : la requête « Je veux une recette qui etc. » sera interprétée sous la forme « Je veux une recette dont la représentation en OWL DL soit une instance de la classe R , où $R \equiv \text{etc.}$ ». Une telle requête est le point de départ de la remémoration et de l'adaptation et constitue donc un problème cible, autrement dit : un problème cible est une classe de recettes : $\models_{\mathcal{O}} \text{tgt} \sqsubseteq \text{Recipe}$.

Hypothèses sur l'annotation. L'annotation est supposée :

- Associer à la recette dans sa globalité un ensemble de classes de recettes auxquelles elle appartient (**ChineseRecipe**, **SaladRecipe**, etc.) ;
- Associer à la recette des ingrédients par la propriété **hasIngredient** ;

⁶On pourrait qualifier ces connaissances de connaissances destinées à l'humain, dans la mesure où les manipulations automatiques sur ces connaissances ne sont pas des raisonnements.

- Associer à chaque ingrédient (e.g., « 30 g. de sucre ») un type d'aliment (« sucre ») et une quantité (« 30 g. ») grâce, respectivement, aux propriétés `hasFood` et `hasQuantity`;
- Identifier dans la recette (titre et préparation) les occurrences des ingrédients;
- Indiquer explicitement le nombre d'ingrédients de la recette, et indiquer que tous les ingrédients donnés sont distincts⁷

Élaboration interactive de `tgt`. Une requête `tgt` est donc une sous-classe de `Recipe`. On supposera qu'elle permet d'indiquer :

- Les types de recettes (des sous-classes atomiques de `Recipe`);
- Les types de recettes à éviter (d'autres sous-classes atomiques de `Recipe`);
- Les types d'aliments requis (des sous-classes atomiques de `Food`);
- Les types d'aliments à éviter (d'autres sous-classes atomiques de `Food`).

Par exemple, la requête suivante

$$\text{tgt} \equiv \text{DesertRecipe} \sqcap \text{ChineseRecipe} \sqcap \text{SaladRecipe} \sqcap \neg \text{CantoneseRecipe} \\ \sqcap \exists \text{hasIngredient}.\exists \text{hasFood}.\text{Ginger} \sqcap \neg \exists \text{hasIngredient}.\exists \text{hasFood}.\text{AlcoholicFood}$$

traduit la requête (exprimée en français) : « Je voudrais une recette de dessert qui soit aussi une salade, d'origine chinoise mais pas cantonnaise, avec du gingembre mais avec aucun ingrédient alcoolique.⁸ »

Dans le concours CCC, il y a trois défis : le défi obligatoire, le *negation challenge* (requête avec des négations) et le défi sur les menus. *A priori*, TAAABLE ne s'attaquera qu'aux deux premiers types de défi⁹.

Comment passer d'une requête textuelle à une requête sous ce format ? Cette question est encore largement ouverte.

Remémoration par classifications dure et élastique. Les procédures de remémoration et d'adaptation présentées ci-dessous sont inspirées de [8].

La classification dure consiste à chercher les instances `srce-case` de la recette `tgt` en utilisant le test d'instanciation. Si une telle instance existe dans la base de cas, elle constitue une solution au problème posé (en espérant qu'il soit bien posé!) et ne nécessite pas d'adaptation. S'il en existe plusieurs, elles pourront être toutes données... En revanche, s'il n'en existe aucune, la classification dure échoue et laisse la place à la classification élastique.

Celle-ci consiste à chercher une modification minimale du problème `tgt` telle que la classification dure n'échoue pas avec ce problème cible modifié. Plus précisément, on suppose qu'on peut définir une suite $(\varphi_n(\text{tgt}))_n$ de sous-classes de `Recipe` telles que $\varphi_0(\text{tgt}) = \text{tgt}$ et $\varphi_{n+1}(\text{tgt})$ est une modification de `tgt` plus « grande » (au sens d'un critère à définir) que $\varphi_n(\text{tgt})$.

Pour spécifier davantage la classification élastique, il faut spécifier les fonctions de modification φ_n . Nous supposons dans la suite que ces fonctions de modification consistent à appliquer une ou plusieurs substitutions choisies parmi les types de substitution suivants :

- Substitution d'une classe de recettes requises par une classe de recettes plus générale (p. ex., substitution de `ChineseRecipe` par `AsianRecipe` ou substitution de `DesertRecipe` par `Recipe`, ce qui revient à supprimer `DesertRecipe` de la requête);
- Substitution d'une classe de recettes à éviter par une classe de recettes plus spécifique (remplacer, par exemple `CantoneseRecipe` par \perp , et donc, $\neg \text{CantoneseRecipe}$ par $\neg \perp$ qui équivaut à \top);

⁷Cela est utile pour affirmer que les connaissances sur la recette sont *complètes*. Cela est utile, par exemple, pour pouvoir inférer qu'une recette est végétarienne (tous les ingrédients connus sont végétariens, et il n'y a aucun autre ingrédient) et plus généralement pour pouvoir traiter les ingrédients interdits dans une requête.

⁸Un aliment alcoolique est-il un aliment qui absorbe de façon régulière des doses excessives d'alcool?

⁹Concernant le troisième, il consiste à faire un menu (donc trois recettes) en prenant comme ingrédients des éléments d'une liste donnée (le contenu d'un garde-manger). On pourrait envisager des requêtes d'une forme plus générale, intégrant ce type de contraintes. Cependant, ce ne serait pas de la tarte, ce qui nous ferait quitter le domaine culinaire.

- (c) Substitution d'un type d'aliment requis par un autre, plus général (p. ex., substitution de `OliveOil` par `Oil`);
- (d) Substitution d'un type d'aliment requis par un autre, connu pour être « culinairement équivalent » (`Butter` par `Margarine`);
- (e) Substitution dans la requête d'un type d'aliment à éviter par un type d'aliment à éviter plus spécifique (p. ex., remplacer `AlcoholicFood` par \perp , ce qui revient à remplacer $\neg\exists\text{hasIngredient}.\exists\text{hasFood}.\text{AlcoholicFood}$ par $\neg\exists\text{hasIngredient}.\exists\text{hasFood}.\perp$ qui équivaut à \top).

Notons que, mis à part les substitutions de type (d), ces substitutions σ constituent des généralisations : $\models_{\mathcal{O}} \text{tgt} \sqsubseteq \sigma(\text{tgt})$.

Une fonction de modification φ_n est donc la composition de substitutions. On supposera qu'à chacune de ces substitutions σ est associé un coût numérique $\text{coût}(\sigma) > 0$ et on définira $\text{coût}(\varphi_n)$ comme étant la somme des $\text{coût}(\sigma)$ pour σ : substitution composant φ_n . Ce coût permettra d'ordonner les modifications par coût croissant, autrement dit, on supposera que $\text{coût}(\varphi_n) \leq \text{coût}(\varphi_{n+1})$ pour tout n .

Adaptation. Si la remémoration n'échoue pas, elle conduit à $\varphi_n(\text{tgt})$ (avec $n = 0$ ssi la classification dure n'a pas échoué) classes de recettes dont au moins un cas source `srce-case` est instance : `srce-case` est une recette répondant à la requête $\varphi_n(\text{tgt})$. Par ailleurs, φ_n est la composition de q opérations de substitutions $\sigma_1, \sigma_2, \dots, \sigma_q$. Soit $\text{pb}_0 = \varphi_n(\text{tgt}) = \sigma_q \circ \sigma_{q-1} \circ \dots \circ \sigma_1(\text{tgt})$, $\text{pb}_1 = \sigma_{q-1} \circ \dots \circ \sigma_1(\text{tgt})$, ..., $\text{pb}_{q-1} = \sigma_1(\text{tgt})$ et $\text{pb}_q = \text{tgt}$. On a alors le chemin de similarité suivant :

$$\varphi_n(\text{tgt}) = \text{pb}_0 \xleftarrow{\sigma_q} \text{pb}_1 \xleftarrow{\sigma_{q-1}} \text{pb}_2 \dots \text{pb}_{q-1} \xleftarrow{\sigma_1} \text{pb}_q = \text{tgt}$$

$\text{Sol}(\text{pb}_0) = \text{Sol}(\varphi_n(\text{tgt})) = \text{srce-case}$ est la recette remémorée associée à pb_0 . Adapter cette recette consiste à « suivre » le chemin de similarité dans l'espace des problèmes, i.e., à

- (1) Adapter $\text{Sol}(\text{pb}_0)$ en une solution $\text{Sol}(\text{pb}_1)$ de pb_1 ,
- (2) Adapter $\text{Sol}(\text{pb}_1)$ en une solution $\text{Sol}(\text{pb}_2)$ de pb_2 ,
- ...
- (q) Adapter $\text{Sol}(\text{pb}_{q-1})$ en une solution $\text{Sol}(\text{pb}_q)$ de $\text{pb}_q = \text{tgt}$.

L'étape d'adaptation numéro i dépend du type de substitution de σ_{q-i+1} . Si ce type est (d), l'adaptation de $\text{Sol}(\text{pb}_{i-1})$ en $\text{Sol}(\text{pb}_i)$ consiste simplement à faire la substitution inverse σ_{q-i+1}^{-1} (p. ex., `Margarine` en `Butter`) sur $\text{Sol}(\text{pb}_{i-1})$.

Si ce type de solution est (a), (b), (c) ou (e), σ_{q-i+1} consiste à faire une généralisation. Autrement dit, la requête pb_{i-1} est moins contrainte que la requête pb_i . L'adaptation envisagée de $\text{Sol}(\text{pb}_{i-1})$ en $\text{Sol}(\text{pb}_i)$ pourrait s'appuyer sur les principes de l'adaptation conservatrice [7]. Ce point reste à étudier en détail.

Pour les substitutions d'aliments, il faut non seulement substituer le type d'aliment, mais aussi la quantité associée. Nous faisons l'hypothèse qu'une substitution doit se faire à masse constante¹⁰. Par conséquent, s'il faut substituer 3 gousses d'ail par x cuillère à soupe d'huile d'olives¹¹, on prendra $x = 3 \times \frac{m_{\text{gda}}}{m_{\text{csho}}}$ où m_{gda} est la masse d'une gousse d'ail et m_{csho} est celle d'une cuillère à soupe d'huile d'olives. Il est donc nécessaire, pour effectuer ce genre d'ajustement des quantités de connaître la masse unitaire (pour quelle(s) unité(s)?) d'un type d'aliment.

Présentation du cas en sortie. Le résultat de l'adaptation est donc une instance $\text{Sol}(\text{tgt})$ de la classe `Recipe`. Par ailleurs, on peut avoir retenu les étapes du processus d'adaptation, qui permettent de définir cette solution à partir de $\text{Sol}(\text{pb}_0) = \text{srce-case}$. Sur la base de cette représentation du processus d'adaptation et sur le cas textuel annoté associé à `srce-case`, on peut construire un cas textuel annoté à fournir en sortie du système de R&PC.

¹⁰Une adaptation plus sophistiquée tiendrait également compte des volumes ou de l'apport nutritif.

¹¹Pourquoi pas?

4 Ontologie culinaire

4.1 Principes de construction

Le principe de construction adopté pour l'ontologie POTAAABLE respecte le processus de construction commun à l'ensemble des méthodologies existantes [4] : étude de faisabilité, analyse des besoins, conceptualisation, maintenance et évaluation. Dans ce papier, nous nous situons dans la phase de conceptualisation.

L'étude des recettes a conduit à un modèle de représentation des connaissances du domaine de la cuisine qui fait référence à de grandes classes de connaissances telles que : recette, ingrédients/aliments, actions à réaliser, quantités et ustensiles utiles à la réalisation de la recette. L'examen de la recette présentée figure 1, met en évidence les relations liant les différentes classes. Une recette est organisée en phases. Chaque phase est ordonnée par des actions. Une tâche est composée d'une action, une durée et comporte des informations sur les ingrédients entrant dans la réalisation de la tâche et la nature du produit résultant de cette tâche. Ces classes de connaissances sont suffisamment indépendantes pour envisager une représentation modulaire du domaine. Nous avons par conséquent adopté le principe de la construction d'une ontologie modulaire. Les principaux modules sont : **Recipe**, **Ingredient**, **FoodComponent**, **Ustensil**. L'avantage de cette organisation modulaire est qu'elle laisse en principe la possibilité d'enrichir les différents modules (en raison de leur indépendance) sans remettre en cause l'organisation générale des modules principaux de l'ontologie. Les relations liant les modules ont été établies manuellement à partir de l'étude globale de la structure des recettes. La question de la réutilisation de ressources ontologiques existantes a été examinée (plusieurs ontologies du domaine existent [9, 1]). Cette organisation permet aussi la répartition des différents modules entre les différents partenaires au projet en utilisant au mieux leur compétences. Ainsi, le module ingrédient est construit en utilisant des méthodes de fouille de textes et l'exploitation du *cook's thesaurus*¹². La granularité des concepts définis dans le module **Ustensil** sera faible par rapport à celui des concepts du module **Ingredient** dans la mesure où l'adaptation porte essentiellement sur les concepts de ce dernier.

Le principe du système de RÀPC de TAAABLE présenté ci-dessus a des conséquences directes sur la construction de l'ontologie POTAAABLE. Tout d'abord, il nécessite l'introduction d'un certain nombre de classes (**Recipe**, **ChineseRecipe**, **SaladRecipe**, **Ginger**, etc.) et de propriétés (**hasIngredient**, **hasQuantity**, etc.). Ensuite, il oriente les priorités sur la construction de l'ontologie, d'une part à cause du raisonnement en lui-même, d'autre part, à cause de l'activité d'annotation préalable. Ainsi, même si les notions culinaires générales telles que « aliment » ou « ustensile de cuisine » devront être représentées dans POTAAABLE, les sous-classes d'aliments (sous-classes de **Food**) seront, au moins dans un premier temps, bien plus détaillées que les sous-classes d'ustensiles de cuisine et cela, parce que les adaptations envisagées ne porteront que sur les ingrédients.

4.2 Résultat

Les figures 2 et 3 présentent une ontologie du domaine culinaire :

- La partie (a) introduit les classes parmi les plus générales, à savoir toutes les classes atomiques descendantes directes de \top et certaines de celles qui sont directement au-dessus de celles-ci. Dire que C est une sous-classe de D , c'est introduire la formule $C \sqsubseteq D$.
- La partie (b) introduit les propriétés liant ces classes. La distinction entre propriétés d'objets et propriétés de types de données peut se faire grâce au co-domaine. Dire que p est une sous-propriété de q , c'est introduire la formule $p \sqsubseteq q$. La colonne « ? » permet d'introduire les axiomes de type « p est fonctionnel » (lettre f) et de type « p est transitif » (lettre t).
- La partie (c) donne quelques assertions. En particulier, elle permet d'introduire l'unité **unitaryUnit** qui est « l'unité des choses qui n'en ont pas ». Si l'unité de « 300 g de farine » est **gramme** (de symbole "g"), l'unité de « 4 abricots » est **unitaryUnit** (dont le symbole est "").

¹²<http://www.foodsubs.com>

- La partie (d) illustre graphiquement une partie des classes et des propriétés présentées en (a) et (b). Une recette (*Recipe*) décrit la façon d’obtenir un plat (*Dish*) comprenant un ensemble de composants (*FoodComponent*). Une description de recette débute par une liste des composants nécessaires à sa réalisation associés à des quantités et à un mode opératoire. Les composants peuvent être soit des ingrédients de base (*Ingredient*) soit des préparations (*Dish*). Ils associent un aliment (*Food*) avec une quantité (*Quantity*) qui est caractérisée par une valeur et une unité.
- La partie (e) illustre une autre partie des classes et des propriétés présentées en (a) et (b). Le mode opératoire permettant d’obtenir le plat est décrit à l’aide d’un ensemble d’actions (*RecipeAction*) manipulant des composants. Une action élémentaire (*RecipeSimpleAction*) est associée à un ensemble de paramètres d’actions précisant la façon de réaliser l’action.
- La partie (f) présente un extrait de l’ontologie des actions culinaires.

4.3 Annotation d’un cas textuel

L’annotation partielle du cas de la figure 1 est indiquée à la figure 4.

5 Lien entre la construction de la partie haute de l’ontologie et sa spécialisation

La partie haute de l’ontologie POTAAABLE a été conçue manuellement. Ce choix découle de la volonté d’adapter cette ontologie aux exigences de l’adaptation en RÀPC présentées plus haut. Mais il a pour conséquence l’impossibilité pratique de définir le grand nombre de classes nécessaires à la représentation des recettes qui constituent la base de cas textuels. Par exemple, si l’on peut envisager de créer la classe **Food**, ainsi que ses principales sous-classes (par exemple **Vegetables** ou **Fruit**), il n’est pas envisageable de créer exhaustivement toutes les sous-classes de ces dernières.

Dans la mesure où nous devons, dans le cadre du projet TAAABLE, avoir recours à des techniques de traitement automatique de la langue pour annoter les cas textuels (le peuplement de l’ontologie), nous avons pris le parti d’utiliser ces techniques également pour générer la partie « basse » de l’ontologie, c’est-à-dire l’ensemble des nombreuses sous-classes des classes définies manuellement, pratique courante en ingénierie des connaissances. Cette section discute de cette étape, baptisée *spécialisation* de l’ontologie dans la mesure où elle consiste à générer des classes plus spécifiques que celle de la partie haute.

Remarquons tout d’abord que ces étapes ne sont pas chronologiquement aussi bien ordonnées, et ne peuvent en général pas l’être : les connaissances abstraites sur le domaine proviennent le plus souvent d’une généralisation des connaissances concrètes plus spécifiques. La conception de POTAAABLE est donc partie d’un *a priori* inévitable sur ce que sera sa partie basse, les classes proposées ont à tout moment été confrontées à des recettes de la base. Mais l’ontologie haute reflète aussi l’intention de ses concepteurs, au regard, dans le cas de cette application, des mécanismes d’adaptation qu’elle va permettre, et qui conditionnent l’engagement ontologique [5]. Il faut par exemple trancher sur la question classique de la classification de la tomate, et dans le cas qui nous intéresse, on choisit de sacrifier la réalité botanique ($\text{Tomato} \sqsubseteq \text{Fruit}$) à la pragmatique, qui veut qu’on substitue plus volontiers un légume qu’un fruit à la tomate. Une autre façon d’aborder ce problème est de considérer que l’on a deux points de vue : celui de la botanique et celui de la cuisine. Ainsi, on aurait les axiomes $\text{Cooking} : \text{Tomato} \sqsubseteq \text{Cooking} : \text{Vegetable}$ et $\text{Botanic} : \text{Tomato} \sqsubseteq \text{Botanic} : \text{Root}$.

Il reste à déterminer dans quelle mesure la spécialisation automatique de notre ontologie (1) sera influencée par les choix cristallisés dans sa partie haute, et (2) remettra éventuellement en cause certains de ces choix. Les techniques de TAL utilisées pour l’aide à la conception d’ontologie doivent en effet être capables d’agrèger les concepts identifiés en concepts plus généraux, en fonction de la similarité entre les contextes d’utilisation des termes dénotant les concepts spécifiques.

Nom de la classe	Sous-classe de	Explication
Recipe	⊤	Classe des recettes
FoodComponent	⊤	Une instance de cette classe représente un aliment (et sa quantité), qu'il soit en début de recette (cf. Ingredient), en fin (cf. Dish) ou au milieu.
Ingredient	FoodComponent	Ingrédient
Dish	FoodComponent	Plat
Food	⊤	Nourriture sans quantité associée
Quantity	⊤	Quantité de nourriture
Unit	⊤	Unité
MassUnit	Unit	Unité de masse
TemperatureUnit	Unit	Unité de température
TimeUnit	Unit	Unité de temps
VolumeUnit	Unit	Unité de volume
IngredientFood	Food	Nourriture susceptible de composer un ingrédient
RecipeAction	⊤	Une action de la recette.
RecipeSimpleAction	RecipeAction	Action élémentaire.
RecipeComplexAction	RecipeAction	Action composée d'actions élémentaires.
ActionParameter	⊤	Ensemble des paramètres d'une action (hors des FoodComponent)
ActionQualifier	⊤	Qualificatif de l'action (correspond aux adverbes)
Ustensil	⊤	Ustensile de cuisine (four, récipient, spatule...)
StopCondition	⊤	Condition d'arrêt d'une action
Until	StopCondition	Condition d'arrêt de type « jusqu'à ce que ... »
During	StopCondition	Condition d'arrêt de type durée
Duration	Quantity	Durée, qui est une quantité dont l'unité doit être une unité de temps, ce qui s'écrit : $Duration \sqsubseteq \exists hasUnit.TimeUnit$.

(a) Classes atomiques générales (premier niveau et partie du deuxième).

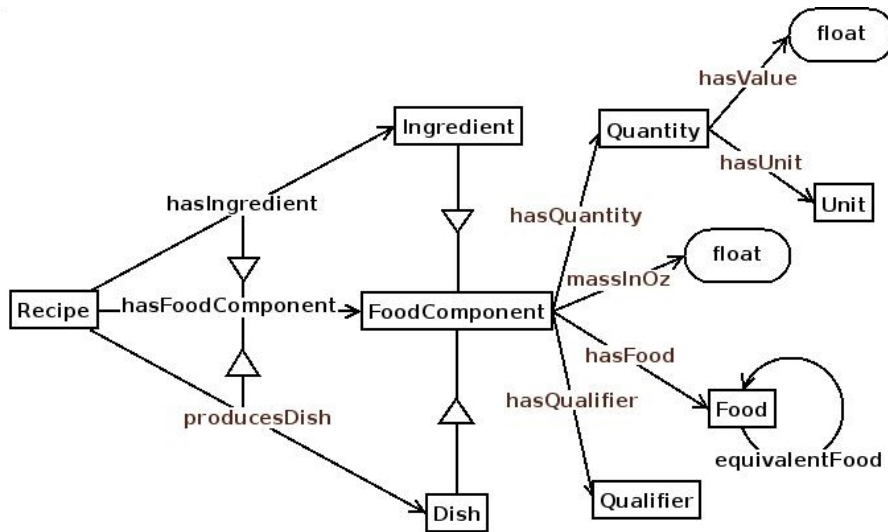
Nom	Domaine	Co-domaine	Sous-propriété de	?
hasFoodComponent	Recipe	FoodComponent		
hasIngredient	Recipe	Ingredient	hasFoodComponent	
producesDish	Recipe	Dish	hasFoodComponent	f
massInOz	FoodComponent	float		f
hasQuantity	FoodComponent	Quantity		f
hasFood	FoodComponent	Food		f
hasIngredientFood	FoodComponent	IngredientFood		f
hasValue	Quantity	float		f
hasUnit	Quantity	Unit		f
hasUnitSymbol	Unit	string		f
hasInputFoodComponent	RecipeAction	FoodComponent		
hasOutputFoodComponent	RecipeAction	FoodComponent		
hasActionParameter	RecipeSimpleAction	ActionParameter		f
hasStopCondition	ActionParameter	StopCondition		
hasUntilText	Until	string		f

(b) Propriétés d'objets et propriétés de types de données.

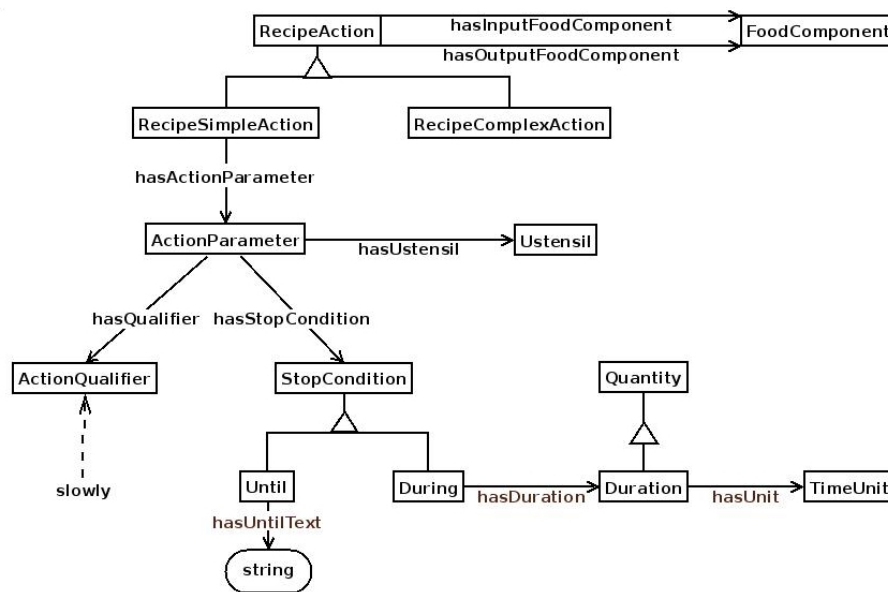
MassUnit(gramme)	hUS(gramme, "g")	MassUnit(ounce)	hUS(ounce, "oz")
TemperatureUnit(celcius)	hUS(celcius, "C")	TemperatureUnit(fahrenheit)	hUS(fahrenheit, "F")
TimeUnit(minute)	hUS(minute, "mn")	TimeUnit(second)	hUS(second, "s")
VolumeUnit(litre)	hUS(litre, "l")	VolumeUnit(tablespoon)	hUS(tablespoon, "tsp")
Unit(unitaryUnit)	hUS(unitaryUnit, "")	ActionQualifier(slowly)	
Until(until1)	hasUntilText(until1, "until the meat is very tender")		

(c) Quelques assertions (hUS est une abréviation de hasUnitSymbol).

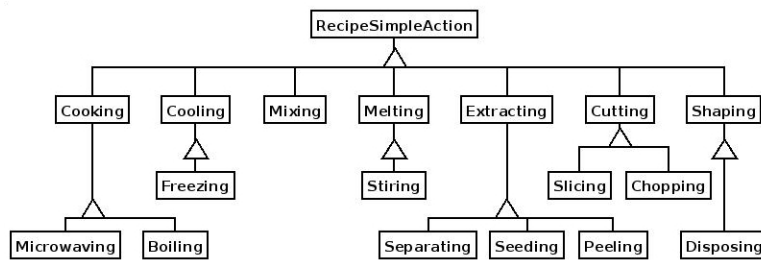
FIG. 2 – Le haut d'ontologie de TAAABLE (1/2).



(d) Illustration des classes et des propriétés relatives à la description d'une recette `RecipeSimpleAction`.



(e) Illustration des classes et des propriétés relatives aux actions culinaires que `RecipeSimpleAction`



(f) Extrait de l'ontologie des actions culinaires

FIG. 3 – Le haut d'ontologie de TAAABLE (2/2).

<RECIPE>

<TI>Citrus Marinade and Salsa (for Chicken or Fish)</TI>

<IN>1/4 c Lemon juice</IN>

Ingredient(ing1) hasFood(ing1,lj1) Juice(lj1) hasQualifier(lj1,qual1)
ObtainedFromFood(qual1) obtainedFrom(qual1,lm1) Lemon(lm1) hasQuantity(ing1,quant1)
hasValue(quant1,0.25) hasUnit(quant1,cup)

<IN>2 tb Dry sherry</IN>

Ingredient(ing2) hasFood(ing2,ds1) DrySherry(ds1) hasQuantity(ing2,quant2)
hasValue(quant2,0.25) hasUnit(quant2,tablespoon)

<IN>2 tb Olive oil or cooking oil</IN>

Ingredient(ing3) hasFood(ing3,oooco) (OliveOil|CookingOil)(oooco)
hasQuantity(ing3,quant3) hasValue(quant3,2.) hasUnit(quant3,tablespoon)

<IN>1/2 ts Dried oregano, crushed</IN>

<IN>1/4 ts Garlic salt</IN>

<IN>1 md Tomato, peeled, seeded and chopped</IN>

Ingredient(ing6) hasFood(ing6,tom1) hasQualifier(tom1,medium) hasQualifier(tom1,qt1)
ingOfEd(qt1,at1) Peeling(at1) hasQualifier(tom1,qt2) ingOfEd(qt2,at2) Seeding(at3)
hasQualifier(tom1,qt3) ingOfEd(qt3,at3) Chopping(at3) hasQuantity(ing6,quant6)
hasValue(quant6,1.) hasUnit(quant6,unitaryUnit)

<IN>1 Green onion, sliced</IN>

<IN>1/4 c Sliced pitted ripe olives</IN>

<IN>1 tb Snipped parsley</IN>

<IN>1 tb Slivered almonds</IN>

<PR>

For marinade, in a 1cup measure combine lemon juice, sherry, oil, oregano,
and garlic salt.

Combining(a1) hasInputFoodComponent(a1,ing1) hasInputFoodComponent(a1,ing2)
hasInputFoodComponent(a1,ing3) hasInputFoodComponent(a1,ing4)
hasInputFoodComponent(a1,ing5) hasOutputFoodComponent(a1,fc1) hasActionParameter(a1,ap1)
hasUstensil(ap1,u1) Measure(u1) hasVolume(u1,1cup) hasValue(1cup,1.) hasUnit(1cup,c)

Set aside 2 tablespoons. Pour remaining into a plastic bag in a bowl; add
chicken or fish. Close bag; chill 1 hour, turning once. For salsa, combine
the 2 tablespoons marinade, tomato, green onion, olives, parsley, and
almonds. Cover; chill till serving time. Drain chicken or fish, reserving
marinade. Grill till chicken is tender and fish is flaky, turning and brushing
often with the marinade. Serve with chilled salsa and avocado slices, if desired.
Makes 4 servings. Nutrition facts per tablespoon salsa: 13 cal.

</PR>

</RECIPE>

FIG. 4 – Un cas textuel du CCC partiellement annoté.

6 Conclusion et perspectives

Un système de raisonnement à partir de cas utilise plusieurs bases de connaissances, en particulier les cas, les connaissances d'adaptation et les connaissances du domaine. Ces dernières se présentent souvent sous la forme de classes (ou concepts) utiles pour représenter et organiser les cas, et de liens entre ces classes (liens de subsumption, de composition, etc.) ; elles peuvent donc prendre la forme d'une ontologie. Le système de RÀPC TAAABLE est en cours de conception. Il a pour objectif la résolution de problèmes culinaires et s'appuie sur l'ontologie POTAAABLE qui est en cours de développement et utilisera le langage OWL-DL. Cet article présente le travail de conception de la partie de POTAAABLE contenant les concepts culinaires généraux (recettes, aliments, actions culinaires, etc.) et leurs liens. Bien que cette ontologie se veuille partageable et réutilisable pour d'autres applications dans le domaine culinaire, les priorités dans son développement sont dictées par les besoins du système TAAABLE. En particulier, l'adaptation devrait essentiellement se faire par substitutions de types d'aliments, et donc le concept de type d'aliment et les concepts « connexes » ont une priorité importante lors du développement de l'ontologie.

Par ailleurs, l'ontologie actuelle ne prend pas encore en compte tous les éléments qui pourraient être utiles :

- Goût (salé, sucré, acide, amer, etc.), odeur, consistance des aliments
- Rôle d'un aliment dans un plat (protéine, légume, féculent, etc.)
- Possibilité de combinaison d'aliments
- Équivalences entre aliments

En parallèle avec ce travail ont lieu plusieurs travaux dans le cadre du projet TAAABLE :

- Exploitation semi-automatique de sites Web spécialisés dans la cuisine ;
- Travail sur les concepts plus spécifiques de l'ontologie et sur leurs liens utilisant des techniques de fouille de textes (sur la base des recettes du CCC, notamment) ;
- Travail sur les types de recettes en vue de leur indexation par des classes de recettes :
 - Types « géographiques » : travail visant à associer une région géographique à une recette (recettes chinoises, méditerranéennes, castelthéodoriciennes, etc.) ;
 - Autres types de recettes (entrées, plats de résistance, desserts, salades, soupes, plats végétariens, sans alcool, sans sel, sans goût) ;
- Etc.

Une des perspectives à court terme est l'articulation de ces différents travaux.

D'un point de vue technique, le problème du temps de calcul devra également être abordé. Si on considère l'ensemble de l'ontologie POTAAABLE une fois celle-ci finie et son peuplement par les annotations de recettes (ce qui fait plusieurs instances par recettes), sachant que la base contient plusieurs centaines de recettes¹³, cela fait une base de connaissances énorme et rendra probablement la remémoration trop coûteuse (temps de calcul mais aussi — surtout ? — place mémoire nécessaire). Ce problème pourra être abordé en utilisant, pour la remémoration, une approximation de la base de connaissances dans un langage de représentation moins expressif mais utilisant des inférences polynomiales. Cela pourra être accompagné d'un travail pour une structuration de la base de cas par l'introduction de classes supplémentaires. Pour l'adaptation, seule la recette remémorée et les instances qui y sont attachées seront à mobiliser, le temps de calcul devrait être moins problématique.

Enfin, ce projet est l'occasion pour des chercheurs en RÀPC et aussi dans d'autres domaines des systèmes à base de connaissances (extraction des connaissances, TAL, acquisition/ingénierie des connaissances, ontologies, multi-agents, etc.) de collaborer sur un projet applicatif. Au-delà de cette application, il importe que des leçons générales soient tirées pour l'articulation de ces différents domaines scientifiques.

¹³900 recettes ont été fournies par les organisateurs du CCC début septembre 2007 et de nombreuses autres seront fournies début août 2008.

Références

- [1] F. Batista, J. P. Pardal, P. Vaz, N. Mamede, et R. Ribeiro. Ontology construction : cooking domain. Technical report, INESC-ID, Lisboa, Portugal, 2006.
- [2] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, et L. Andrea Stein. OWL Web Ontology Language Reference. W3C Recommendation, February 2004. www.w3.org/TR/owl-ref.
- [3] S. Begum, M. U. Ahmed, P. Funk, N. Xiong, et B. von Schéele. Similarity of medical cases in health care using cosine similarity and ontology. In *5th Workshop on CBR in the Health Sciences*, pages 263–272, Belfast, Northern Ireland, August 2007. Springer LNCS.
- [4] P. Cimiano, J. Volker, et R. Studer. Ontologies on demand? In *Information Wissenschaft and Praxis*, volume 57, pages 315–320, 2006.
- [5] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2) :199–220, 1993.
- [6] T. R. Gruber. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993.
- [7] J. Lieber. Application of the Revision Theory to Adaptation in Case-Based Reasoning : the Conservative Adaptation. In *Proceedings of the 7th International Conference on Case-Based Reasoning*, Lecture Notes in Artificial Intelligence 4626, pages 239–253. Springer, Belfast, 2007.
- [8] J. Lieber et A. Napoli. Correct and Complete Retrieval for Case-Based Problem-Solving. In Henri Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, Brighton, United Kingdom, pages 68–72, 1998.
- [9] United States Department of Agriculture USDA. <http://www.ffao/agrovoc/>, 2004.
- [10] R. Weber, K. D. Ashley, et S. Brüninghaus. Textual reasoning. In *Twelfth SIGCSE'81*, volume 13, 2005.