Reasoning with Qualitative Spatial and Temporal Textual Cases

Valmi Dufour-Lussier

École doctorale IAEM

Reasoning with Qualitative Spatial and Temporal Textual Cases

thèse

présentée et soutenue publiquement le

7 octobre 2014

pour obtenir le titre de

docteur de l'Université de Lorraine

(spécialité informatique)

par

Valmi Dufour-Lussier

devant le jury composé de

Isabelle Bloch (présidente),	Télécom ParisTech
Claire Gardent (examinatrice),	CNRS Nancy
Florence Le Ber (directrice),	ENGEE Strasbourg
Luc Lamontagne (rapporteur),	Université Laval
Jean Lieber (directeur),	Université de Lorraine
Mirjam Minor (rapporteuse),	Goethe-Universität Frankfurt

École doctorale IAEM

Reasoning with Qualitative Spatial and Temporal Textual Cases

thèse

présentée et soutenue publiquement le

7 octobre 2014

pour obtenir le titre de

docteur de l'Université de Lorraine

(spécialité informatique)

par

Valmi Dufour-Lussier

devant le jury composé de

Isabelle Bloch (présidente),	Télécom ParisTech
Claire Gardent (examinatrice),	CNRS Nancy
Florence Le Ber (directrice),	ENGEE Strasbourg
Luc Lamontagne (rapporteur),	Université Laval
Jean Lieber (directeur),	Université de Lorraine
Mirjam Minor (rapporteuse),	Goethe-Universität Frankfurt

Abstract

This thesis proposes a practical model making it possible to implement a case-based reasoning system that adapts processes represented as natural language text. The use of natural language simplifies both the modelling and the execution by avoiding the need for the users to use special formalisms such as workflows to represent processes. In answer to a query describing a goal, the system shall be able to present the user with a consistent set of instructions enabling them to achieve that goal, expressed using natural language.

In order to make inferences possible, a formal representation of a process ought to be attached to the text describing it. We use classical methods from natural language processing, a custom anaphora resolution mechanism and a set of annotation rules to extract events and objects from instruction texts, as well as temporal constraints represented using a qualitative interval algebra.

During the adaptation stage, substitutions are performed in the source solution in such a way that it becomes a solution to the target problem. Temporal constraints are modified using a belief revision operator in order to maintain consistency with the application domain knowledge. We define two belief revision operators applicable on qualitative algebras: the first, using a best-first search algorithm, is consistent with the Alchourrón, Gärdenfors and Makinson (1985) postulates. The second is a repair propagation algorithm based on Vilain and Kautz (1986). It is faster, but may not obey all the postulates. It is shown that the reasoning process applied to processes can also be applied to different problems, such as farming problems, that can be represented using a qualitative algebra.

Finally, the annotation rules are applied inversely with respect to temporal constraint changes, in a text regeneration stage. This has the effect of making minimal modifications to the text that make it consistent with the new temporal constraints. Strategies are used to maintain global consistency and anaphoric cohesion.

The proposed model was applied to cooking problems, and implemented as a Facebook application, named Craqpot. Comparative tests were run, in which our solution was compared to a retrieval-only solution and a solution performing a more superficial adaptation. Our in-depth adaptation model produced texts of the same quality as the more superficial solution, but the recipes themselves were judged slightly better. The quality of the adapted recipes and texts were expectedly not as good as that of unmodified recipes and texts from the case base. Overall though, the users were as much satisfied with the deeply adapted recipes as with the original ones, and were much less satisfied with the superficially adapted recipes.

Résumé

Cette thèse propose un modèle permettant la mise en œuvre d'un système de raisonnement à partir de cas capable d'adapter des procédures représentées sous forme de texte en langue naturelle. L'utilisation de la langue naturelle simplifie la modélisation et l'exécution en évitant à l'utilisateur de devoir apprendre des formalismes spécialisés, tels que les flux opérationnels (ou *workflows*), utilisés pour représenter des procédures. En réponse à une requête décrivant un but, le système proposé devra être en mesure de présenter à l'utilisateur un ensemble d'instructions lui permettant d'atteindre le but, exprimées en langue naturelle.

Pour permettre les inférences, une représentation formelle de la procédure doit être rattachée au texte qui la décrit. Nous utilisons des méthodes classiques en traitement automatique des langues ainsi qu'un mécanisme sur mesure de résolution des anaphores et un ensemble de règles d'annotations pour extraite des textes d'instructions des événements, des objets et des contraintes temporelles représentées à l'aide d'une algèbre qualitative d'intervalles.

Durant l'étape d'adaptation, on procède à des substitutions dans la solution source, de façon à en faire une solution au problème cible. Pour maintenir la cohérence avec les connaissances du domaine d'application, des contraintes temporelles sont modifiées à l'aide d'un opérateur de révision des croyances. Nous définissons deux opérateurs de révision applicables aux algèbres qualitatives : le premier, utilisant un algorithme du meilleur d'abord (ou *best-first search*), respecte les postulats définis par Alchourrón, Gärdenfors et Makinson (1985). Le second est un algorithme de propagation de réparation inspiré par l'algorithme de Vilain et Kautz (1986). Ce dernier est plus rapide, mais ne respecte pas nécessairement tous les postulats. Il est démontré par ailleurs que l'approche employée pour les procédures peut être appliquée a d'autres types de problèmes pouvant être représentés à l'aide d'algèbres qualitatives, par exemple des problèmes agricoles.

Finalement, lors d'une étape de regénération du texte, les règles d'annotation font l'objet d'une application inverse en fonction des contraintes temporelles modifiées. Cela a pour effet de modifier de façon minimale le texte pour le rendre cohérent avec les nouvelles contraintes. On emploie des stratégies additionnelles pour maintenir la cohérence globale et la cohésion anaphorique.

Le modèle proposé a été appliqué à des problèmes de cuisine et mis en œuvre dans une application Facebook nommée Craqpot. Des tests comparatifs ont été exécutés pour comparer notre application à une solution fondée sur la remémoration simple ou sur une adaptation plus superficielle. Notre modèle d'adaptation en profondeur a généré des textes de la même qualité que la solution plus superficielle, mais les recettes elles-mêmes ont été jugées légèrement meilleures. Comme on peut s'y attendre, la qualité

des textes et des recettes adaptées a été jugée inférieure à celle des textes de recettes non modifiés extraits de la base de cas. Globalement, par contre, les utilisateurs se sont dits autant satisfaits par les recettes adaptées en profondeur que par les recettes originales, mais beaucoup moins satisfaits des recettes adaptées superficiellement.

Resumo

Tiu disertaĵo proponas modelon, ebligante implementar kazbazita rezonado sistemo, kiu adaptas procedurojn reprezentitajn per natura lingvo teksto, en respondo al pridemandoj de uzanto. Dum la kazoj kaj la solvoj estas en teksta formo, la adapto mem estas realigita sur retoj de tempaj limigoj esprimitaj per kvalita algebro, uzanta kredrevizio operatoro. Natura lingvo prilaborado metodoj estas uzitaj por akiri kazo reprezentoj kaj regeneri teksto bazita sur la adapto rezulton.

Summarium

Hoc thesim exemplar proponit quod patefacit constitutionem ratiocinationis casu-substructionis actuare quæ rationes procedendas repræsentatas per linguam naturalem in responsio ad quæsitos adaptat. Cumque casus et solutiones formam textus habent accommodatio ipsa in reticulis de exigentiis temporalibus repræsesentatis per algebram qualitatem cum operatore de recensione opinionum paragitur. Methodi de procedenti linguæ naturalis adhibentur ut repræsentationes casuum acquirant et ex effectum accommodationis textum regenerent.

x

Remerciements

Cette thèse est, d'un certain point de vue, l'aboutissement de trente-trois ans de labeur pour des centaines d'actrices et d'acteurs, voire de trente-trois ans et neuf mois pour au moins l'une d'entre elles. Il me serait bien impossible de mentionner ici le nom de toutes celles et de tous ceux qui ont joué un rôle en vue de faire de moi ce que je suis, en tant que personne, scientifique ou auteur de thèse, tout comme il n'existe pas de mots assez forts pour exprimer tout ce que je dois à certaines personnes. J'espère que ces lacunes me seront pardonnées.

En tout premier lieu, je tiens à exprimer la reconnaissance considérable que je ressens à l'égard de mon directeur, Jean Lieber, pour la somme du temps et des efforts qu'il a investis à me conseiller et à m'aider dans – et hors de – ce travail. Il représente pour moi un mentor, un modèle, un deuxième père et un grand ami.

Je remercie également ma directrice, Florence Le Ber. La distance nous séparant a eu pour effet qu'elle se soit un peu moins impliquée dans cette thèse, ce qui est dommage car ses conseils et nos échanges ont toujours été extrêmement constructifs et enrichissants.

Merci à mon rapporteur, Luc Lamontagne, qui était prêt à venir se perdre jusqu'à Nancy puis a finalement dû gérer les complications techniques d'une vidéoconférence suite à un changement de date. Dank gebührt meiner Zweitgutachterin, Mirjam Minor, die, trotz der Komplexität des französischen Bahnnetzes sowie der französischen Verwaltung, meiner Verteidigung der Dissertation beiwohnen konnte. I thank them both as well for the time and effort they put into reading my thesis, and making numerous thoughtful suggestions to help improve it. Je remercie également Isabelle Bloch d'avoir accepté de présider ce jury, ainsi que Claire Gardent d'avoir accepté d'y participer et d'avoir suivi attentivement mon parcours doctoral à titre de référente.

I would also like to extend my thanks to the national and international case-based reasoning community for being so open and welcoming. In particular, thanks to David Aha for organising the 2012 doctoral consortium, to Ralph Bergmann, Sutanu Chakraborti, Cindy Marling and an additional anonymous reviewer, as well as to all those that participated in the event for their critical appraisal of my research project.

J'adresse également un remerciement tout particulier à Yannick Toussaint et à Bruno Guillaume qui, en acceptant d'encadrer mes stages de master, m'ont initié au monde de la recherche. Merci également à tous ceux qui ont travaillé avec moi ou m'ont offert leur amitié durant mon séjour au LORIA, notamment les membres de l'équipe Orpailleur et son directeur scientifique, Amedeo Napoli, et plus particulièrement à Julien Cojan, Emmanuelle Gaillard, Alice Hermann-Pépin, Laura Martin, Thomas Meilender, Emmanuel Nauer et Guy Perrier, ainsi qu'aux équipes pédagogiques qui m'ont accueilli à bras ouverts à l'IUT Charlemagne ainsi qu'à l'UFR Mathématiques et Informatique.

De nombreuses personnes n'ont pas été impliquées dans mon travail doctorat, mais ont néanmoins joué un rôle capital en m'insufflant la volonté de persévérer dans mes études. J'aimerais tout particulièrement remercier mes anciens professeurs Patrick Blackburn, Victor Boucher et Alain Polguère pour leur générosité et leur passion contagieuse.

Il serait difficile de ne pas mentionner le fait que mes parents, Denise et Marc, m'ont fabriqué, élevé, éduqué, soutenu dans les moments difficile, botté le cul dans les moments de passivité, et aimé, et que sans eux et sans tout cela cette thèse n'aurait jamais vu le jour. À eux, à mon petit frère adoré Samuel et aux autres membres de ma famille, merci pour votre soutien et votre amour.

Mes études ont été semées d'embuches et auraient pu se terminer de façon anticipée n'eut été les encouragements de mes anciennes fiancées, Catherine et Ayla, ainsi que de mon épouse, Roseline. En plus de cuisiner pour moi des risottos et bien d'autre choses encore, Roseline m'a encouragé et soutenu dans mon projet de faire une thèse à Nancy alors que elle-même complétait ses études à Québec, puis elle a mis sa carrière en suspens afin d'être à mes côtés durant les deux dernières années. Merci d'être à mes côtés durant les meilleurs, les pires et les banals moments.

Contents

List of Figures

List of Tables

Résumé long

Riesbeck et Schank (1989) définissent le raisonnement à partir de cas (RàPC) autant comme une hypothèse psychologique sur la nature cognitive de la pensée humaine que comme un cadre de l'intelligence artificielle visant à simuler le processus de la pensée humaine. D'après eux, l'esprit humain ne fonctionne pas en appliquant des règles logiques face aux situations qu'il rencontre. Face à une situation nouvelle, il se remémore plutôt une expérience similaire, un cas, et décide de la réaction appropriée sur la base de cette expérience. Un système de RàPC fonctionne d'une manière similaire : face à une requête énonçant un problème, il remémore un problème passé similaire dont la solution est connue, puis adapte cette solution. On peut donc affirmer que l'intuition sur laquelle se fonde le RàPC est que les problèmes similaires ont des solutions similaires.

Un système de RàPC utilise une base de cas, qui constitue un recueil de problèmes résolus dans le passé, nommés cas, dans un domaine d'application donné, avec leur solution. Chaque fois qu'un nouveau problème, appelé cas cible, est rencontré, la première étape qui doit être entreprise est de récupérer un cas qui pourrait être utilisé pour aider à le résoudre : le cas source. C'est l'étape de remémoration.

Par conséquent, si un cuisiner souhaite obtenir une recette de risotto aux carottes, mais qu'une telle recette n'existe pas, un système de RàPC pourrait lui proposer de prendre et d'adapter une recette similaire, par exemple une recette de risotto aux champignons. Il s'avère qu'il existe des systèmes de ce type. Taaable (Cordier et al., 2013) est l'un d'eux. Il utilise une classification élastique (Lieber, 2002) pour trouver des sources correspondant à une généralisation de la cible : un risotto aux carottes et un risotto aux champignons sont tous les deux des risotto de légumes, donc le risotto aux champignons est remémoré (mais pas le risotto aux saucisses).

Le système doit alors déterminer comment la solution du cas source peut être utilisée pour résoudre le problème cible. C'est l'étape d'adaptation, aussi connue sous le nom de réutilisation. C'est cette étape qui constitue l'objet de cette thèse. Les solutions qui ont été proposées s'inscrivent principalement dans deux catégories :

Adaptation dérivationnelle. Le processus de raisonnement qui a conduit à la construction de la solution du problème source est stocké avec le cas quand celui-ci est ajouté à la base de cas, et est ensuite « rejoué » sur le problème cible (Carbonell, 1985). Le processus peut également être induit. Cette adaptation est également appelée « adaptation générative » (Veloso et al., 1996).

Adaptation transformationnelle. La solution du problème cible est construite à partir de la solution source, modifiée en fonction des différences qui existent entre les problèmes source et cible (Carbonell, 1983).

Une approche courante de la tâche d'adaptation consiste à laisser à l'utilisateur le soin de s'en occuper : la solution du problème source est présentée telle quelle à l'utilisateur, qui décide ou non d'apporter manuellement des ajustements. Cette approche est appelée l'adaptation nulle (ou adaptation par copie) et constitue en quelque sorte un type d'adaptation transformationnelle.

Dans le cas de Taaable, le fait qu'un système de classification hiérarchique soit utilisé fournit gratuitement une première forme d'adaptation : puisque la généralisation a été rendue possible parce que les carottes et les champignons sont deux légumes, alors nous savons qu'une solution au problème « Comment cuisiner un risotto aux carottes ? » pourrait être « préparer un risotto aux champignons avec des carottes au lieu des champignons. ». Si c'est la seule modification apportée à la recette et donc, par exemple, si les carottes ne sont cuites que pendant deux minutes, cela donnera une recette de « risotto aux carottes croquantes ». Par conséquent, la préparation elle-même doit être modifiée pour tenir compte des effets de la substitution de l'ingrédient.

Les problèmes de recettes de cuisine ne constituent qu'un exemple amusant d'une classe de problèmes sérieux. Des procédures d'entreprise *(business processes)*, ou de fabrication, ou encore des protocoles scientifiques sont décrits sous forme de texte, souvent formalisés sous forme de flux opérationnels *(workflows)* pour garantir leur application uniforme et efficace, parfois avec l'aide d'applications logicielles.

Chaque fois que les conditions prises pour acquis lors de la description du processus ne sont pas présentes au moment de son exécution – par exemple, un produit chimique est manquant dans la fabrication d'un médicament pharmaceutique, une procédure administrative a été modifiée, ou un utilisateur veut préparer un risotto aux carottes – l'application devrait être en mesure de mettre à jour le processus à la volée pour aider l'utilisateur à atteindre son but.

La planification classique serait une façon de résoudre ce type de problèmes. Cependant, cela nécessiterait des connaissances de domaine complètes. Par exemple, on devrait disposer de connaissances précises et for-

malisées sur les propriétés physiques des carottes avant et après les avoir découpées, et sur les propriétés idéales que les ingrédients ajoutés à un risotto devraient avoir. Le RàPC, pour sa part, permet de réutiliser des procédures existantes sans avoir la moindre connaissance du domaine, tout en étant capable d'utiliser toute connaissance éventuellement disponible pour améliorer ses résultats.

L'objectif de cette thèse est de proposer un modèle pratique permettant la mise en œuvre d'un système de raisonnement à partir de cas capable d'adapter des procédures dans lesquelles des objets doivent être substitués à ceux qui ont été à l'origine nécessaires. Parce que les cuisiniers ne sont pas des informaticiens et que former tous les utilisateurs d'un système à l'utilisation d'un langage artificiel tel celui des flux opérationnels serait coûteux et peu pratique, notre objectif est de proposer un système qui, pour reprendre les mots de Riesbeck et Schank, donne « des réponses verbales à des problèmes verbaux ».

En réponse à une requête décrivant un but dans un domaine d'application donné, le système décrit ici est en mesure de présenter à l'utilisateur un ensemble cohérent d'instructions lui permettant d'atteindre cet objectif, exprimé en langage naturel. Par exemple, quand un utilisateur demande une de risotto aux carottes, il serait en droit de s'attendre à ce que l'ordre des actions soit modifié afin que tous les ingrédients soient correctement cuits.

Afin de rendre les inférences possibles, cependant, une représentation formelle du processus doit être jointe au texte le décrivant. Nous proposons d'utiliser un formalisme de représentation des connaissances temporelles à cet effet.

Pour réaliser notre objectif, il sera nécessaire d'explorer l'interrelation entre trois domaines de l'intelligence artificielle :

— Le raisonnement à partir de cas permet d'utiliser les connaissances de problèmes passés afin de proposer des solutions hypothétiques à de nouveaux problèmes, à travers la remémoration et l'adaptation.
— Le traitement automatique des langues (TAL) propose des processus permettant de traduire un texte écrit en langue naturelle dans un langage utilisable par une machine, et vice-versa.
— La représentation des connaissances et le raisonnement temporel s'intéressent à des formalismes qui permettent de représenter et de manipuler des connaissances sur des points ou intervalles de temps, et donc également sur les actions.

Chacun de ces domaines rassemble, individuellement, une communauté de recherche importante, et leurs interfaces deux à deux ont également fait l'objet de recherches.

La communauté RàPC reconnaît qu'une partie significative des connaissances sont disponibles dans un format textuel et que, par conséquent, le fait d'être en mesure d'exploiter le texte peut être d'une grande aide dans le déploiement d'un système. La langue naturelle est particulièrement impor-

tante dans certains domaines d'application, tels que la réponse aux questions. Par conséquent, les contributions éventuelles du TAL au RàPC représente un grand intérêt depuis les débuts. Cet intérêt a été exprimé notamment par une série d'ateliers sur le RàPC textuel lors de la conférence AAAI en 1998 ainsi que lors des conférences européennes et internationales sur le RàPC de 2005 à 2007.

La communauté TAL, pour sa part, a toujours exprimé un grand intérêt pour la représentation des connaissances temporelle, et a fortement contribué à ce domaine. En effet, l'algèbre des intervalles de Allen (1983), sans doute l'un des formalismes temporels les plus connus, a été conçu en premier lieu pour représenter des informations temporelles sur des actions décrites en langage naturel. L'annotation temporelle des textes, en particulier, a fait l'objet de beaucoup de travail, ainsi que d'un effort substantiel de normalisation par Pustejovsky et al. (2003).

L'analyse ou la prévision des processus temporels (par exemple le diagnostic) est un sujet d'intérêt pour le RàPC, et a permis des avancées dans le RàPC temporel, la temporalité s'exprimant le plus souvent sous forme de séquences d'événements, éventuellement enrichis d'estampilles temporelles absolues ou relatives. L'adaptation de cas temporels est aussi un problème de recherche important du RàPC sur les processus.

Par contre, les exigences liées à la mise en œuvre d'un système fondé sur ces trois domaines a reçu, au meilleur de notre connaissance, très peu d'attention jusqu'à présent. Par exemple, nous n'avons pas connaissance de travaux ayant précédemment tenté de fournir un modèle intégré de RàPC sur des processus exprimés sous forme de texte et fournissant également des réponses sous forme de texte.

L'évaluation d'un système de RàPC est le plus souvent effectuée empiriquement par des utilisateurs, ce qui rend nécessaire le recutement d'experts du domaine. Un domaine commun tel que la cuisine, en apparence de peu d'intérêt, présente un avantage majeur : trouver des experts ou des quasi-experts est très facile. Par conséquent, il est possible d'évaluer et de comparer à grande échelle de nombreux systèmes différents. C'est pourquoi ce domaine a été choisi comme domaine d'application commun lors des conférences européennes et internationales sur le RàPC afin d'organiser un concours, le *Computer Cooking Contest* (CCC) qui permet de comparer des systèmes développés par des équipes différentes, originalement dans des domaines d'application différents. Les cas dans tous ces systèmes sont des recettes, qui ont une nature textuelle et procédurale. Il n'est donc pas surprenant que la plupart des projets existants visant des objectifs semblables aux nôtres sont également candidats au CCC. Malheureusement, CAKE (Minor et al, 2010), sans doute le système le plus avancé en termes d'adaptation de procédures, présente des limites sur la façon dont il gère le texte. CookIIS (Newo et al, 2010.) est, à notre connaissance, le seul système autre que Taaable qui fournit des réponses textuelles adaptées. À cause de choix

technologiques effectués dans ces deux projets, nous ne pensons pas qu'il soit possible de simplement les fusionner afin d'obtenir la fonctionnalité des deux.

Un effort important a été accompli pour permettre l'adaptation des procédures dans la plateforme CAKE, un système intégré de gestion des procédures et des connaissances. Les flux opérationnels sont utilisés pour représenter les cas procéduraux, plus précisément les flux opérationnels dits « agiles », comme définis par Weber et Wild (2005). Tout comme dans l'application que nous proposons, ceux-ci peuvent être modifiés automatiquement à la volée, suivant des contraintes qui n'avaient pas été prises en compte lors de la création du flux. Pour le CCC, CAKE a été déployé sous le nom CookingCakeWf (Fuchs et al., 2009). Il a ensuite été étendu par Walter et al. (2011) et Schumacher et al. (2014) afin de permettre l'acquisition automatique de cas à partir de texte, en utilisant des techniques d'extraction d'information dans le cas de Walter. Malheureusement, le système ne permet pas d'offrir des réponses textuelles. De plus, nous croyons que l'utilisation de l'extraction d'information impose à CookingCakeWf des limites à la fois en termes d'efficacité et de généricité.

À notre connaissance, le seul système de RàPC procédural qui retourne une réponse textuelle est CookIIS, un autre concurrent dans le CCC. L'étape d'adaptation textuelle est cependant d'une portée limitée. Elle consiste en un algorithme de substitution de chaînes de caractères sophistiqué, appliqué sur le nom des ingrédients remplacés. Le processus lui-même n'est pas adapté – par exemple, si le système adapte une recette en remplaçant du jambon par du bacon, il ne suggérera pas pour autant de faire cuire le bacon. Nous pensons que ceci constitue un problème, et nous souhaitons que notre système soit en mesure de mettre en application des connaissances du domaine telles que « le bacon doit être cuit avant d'être mangé » afin d'adapter le processus d'une manière cohérente.

Nous faisons deux choix techniques, qui seront défendus tout au long de cette thèse, concernant la solution qui conviendrait le mieux à la construction d'un système fonctionnant aussi bien pour l'adaptation procédurale que textuelle. Deux objectifs secondaires informent les choix que nous faisons :

— Maximiser la généricité de notre solution, afin de la rendre utilisable dans d'autres domaines d'application que les procédures – essentiellement, dans la plupart des domaines impliquant le raisonnement spatio-temporel, ce qui justifie le titre de la thèse.
— Maximiser la réutilisation de solutions techniques existantes.

Nous allons d'abord arguer que, bien que des approches de traitement de la langue dites « de surface », inspirées de la recherche et de l'extraction d'information, soient souvent utilisées en RàPC textuel, les approches « profondes » sont plus appropriées pour créer le genre de représentations de cas hautement structurées, telles les flux opérationnels, requises par le

RàPC procédural. En outre, les textes procéduraux présentent peu de variabilité grammaticale ou structurelle – ils sont principalement constitués de phrases exprimant des actions à exécuter, écrites dans l'ordre dans lequel elles doivent être exécutées. Cela implique que des techniques de TAL symbolique qui offrent parfois des résultats insatisfaisants sur du texte libre peuvent être utilisées avec une plus grande efficacité. Néanmoins, de nombreux avantages du TAL sur l'extraction d'information sont conservés, tel qu'une plus grande généricité. Enfin, les méthodes du TAL symbolique permettent de produire une annotation détaillée des caractéristiques linguistiques dans les textes, ce qui s'avérera nécessaire lors de l'étape du processus d'adaptation qui modifie le texte.

Quant à la représentation des connaissances temporelles, nous allons plaider en faveur de l'utilisation des algèbres qualitatives d'intervalles, et plus particulièrement de l'algèbre INDU proposée par Pujari et al. (1999), afin de représenter des processus. Cela implique que les actions dans les processus sont réifiées comme des intervalles de temps. Cette réification sera également étendue aux états et aux durées – permettant d'éviter les contraintes métriques, et limitant ainsi la complexité algorithmique des tâches de raisonnement. Les avantages à utiliser une algèbre intervalle, plutôt que des flux opérationnels – le formalisme le plus couramment utilisé en RàPC procédural – sont nombreuses :

— L'annotation temporelle utilise généralement des formalismes inspirés des algèbres qualitatives, ce qui nous permet de réutiliser des méthodes et des ressources existantes.
— Les relations temporelles utilisées dans la norme d'annotation TimeML (Pustejovsky et al, 2003) sont inspirées de l'algèbre des intervalles de Allen, tout comme INDU. Cela fait en sorte que nous pouvons utiliser TimeML avec des modifications minimes, ce qui constitue un moyen pratique de conserver le texte et la représentation formelle correspondante étroitement mêlés. Comme nous allons le montrer, ceci est nécessaire afin de permettre l'adaptation du texte.
— Des opérateurs de fusion des croyances ont été proposés pour les algèbres qualitatives, notamment par Condotta et al. (2008, 2009). Cela nous permet de définir des opérateurs de révision des croyances qualitatives, qui à leur tour nous permettent d'employer l'approche de l'adaptation basée sur la révision des croyances proposée par Lieber (2007) et Cojan et Lieber (2008).
— En outre, en utilisant les algèbres qualitatives, on obtient pour avantage supplémentaire que la plupart des résultats présentés dans cette thèse peuvent être utilisés dans d'autres domaines d'application portant sur des connaissances qui peuvent être exprimées à l'aide de ces algèbres, ce qui comprend de nombreuses applications spatiales ou temporelles.

Un cadre méthodologique et des algorithmes sont fournis, permettant

d'annoter des informations temporelles dans les textes de procédure et d'appliquer un processus de régénération pour le texte lorsque l'information temporelle doit être modifiée.

Nos principales contributions dans ce domaine sont les suivantes :

— Fournir un modèle pour la résolution des anaphores évolutives, c'est-à-dire des mots (ou l'absence de mots) dans un texte qui réfèrent à des objets qui n'existent qu'à un moment précis dans le temps, en particulier pour les textes procéduraux.
— Fournir un ensemble de règles pour traduire des ensembles de caractéristiques linguistiques en contraintes exprimées en utilisant une algèbre qualitative temporelle.
— Fournir un cadre méthodologique général qui permet de modifier (regénérer) un texte qui a été annoté avec notre système afin de refléter des contraintes temporelles modifiées.

Ces constributions sont mises en œuvre dans une bibliothèque logicielle, mais sous une forme très spécifique au domaine. Rendre cette bibliothèque plus pertinente en séparant la logique de l'annotation/régénération et les règles constitue un travail futur, de même qu'une étude plus rigoureuse du cadre de régénération proposé.

Étant donné un réseau de contraintes temporelles qualitatives sur les actions exprimé en utilisant une algèbre temporelle, nous montrons ensuite qu'il peut être adapté au sens du RàPC en s'appuyant sur le modèle de l'adaptation par un opérateur de révision des croyances. Pour ce faire, il est nécessaire de formaliser la notion de remplacement dans les réseaux de contraintes qualitatives par une abstraction et un raffinement successifs, et de développer un opérateur de révision des croyances pour les algèbres qualitatives. Nous montrons également que cette approche peut être appliquée à différents types de problèmes qui peuvent être formalisés en utilisant les algèbres qualitatives temporelles ou spatiales, et donnons un exemple d'application dans le domaine de l'agronomie.

Nos principales contributions dans ce domaine sont les suivants :

— Fournir et implémenter un opérateur de révision des croyances pour les algèbres qualitatives qui est conforme aux postulats AGM. Cet opérateur, basé sur les travaux existants sur la fusion de croyances, utilise un algorithme A* pour chercher l'espace de modèle.
— Fournir et mettre en œubre un opérateur de révision des croyances pour les algèbres qualitatives qui ne respecte pas tous les postulats AGM, mais présente des qualités algorithmiques supérieures.

Ce premier opérateur est disponible dans Revisor, une collection libre de moteurs de révision des croyances et d'adaptation. Quelques optimisations possibles ont été identifiées. Étudier les propriétés du deuxième opérateur de révision demeure un travail futur.

Enfin, nous avons réuni les implémentations des différentes parties de notre travail, ajouté un module de gestion de connaissances du domaine, et

fourni une interface web. Nous avons nommé cette application web Craqpot. Cela a permis d'évaluer notre proposition et de la comparer à un système qui utilise la remémoration sans adaptation – donnant des résultats de grande qualité à 46% des requêtes mais aucun résultat aux autres – et à un système qui effectue une adaptation minimale fondée sur une substitution de chaînes.

Les résultats préliminaires sur la base de l'évaluation de 50 requêtes par neuf utilisateurs du système suggèrent que la performance de Craqpot est légèrement inférieure au système de remémoration simple sur la qualité du texte et des recettes, mais supérieure à celle de l'autre système sur ces mêmes critères. Ils montrent également que les utilisateurs pensent que Craqpot répond aux requêtes de façon aussi appropriée que la remémoration, et beaucoup mieux que l'autre système.

Cela nous permet d'affirmer que la présente thèse propose un modèle performant et efficace permettant d'employer le raisonnement à partir de cas afin d'adapter des procédures écrites lorsque les prérequis ne sont pas réunis.

Introduction

Roseline's husband is coming home from a long day of rewriting his thesis's introduction. While they are a modern couple, she wants to be kind and cook the dinner so that it be ready when he arrives. They both like risotto, and there are some carrots in the fridge which are about to wither. Therefore, a carrot risotto seems like a good idea. Roseline reaches for her cookbooks. There are plenty of risotto recipes to be found, but there is no such thing as a carrot risotto recipe.

Introducing Case-Based Reasoning

Problems of this type, in which the solutions to similar problems are known, are tailor-made for the case-based reasoning (CBR) paradigm. Riesbeck and Schank (1989) define CBR as much as a psychological hypothesis about the cognitive nature of human thought as an artificial intelligence framework aiming at simulating the human thinking process. The human mind, they claim, does not normally work by applying rules, logical or otherwise, to situations it encounters. Rather, when faced with a new situation, it retrieves a similar past experience, a *case,* and decides on the proper reaction based on this experience. A CBR system works in a similar fashion: when queried with a problem, it attempts to retrieve a similar problem from the past for which the solution is known and adapt this solution. The intuition behind CBR, therefore, is that similar problems have similar solutions.

Retrieval

A CBR system uses a *case base,* which is a compendium of problems solved in the past, named *cases*, from a given application domain, along with their solution. Whenever a new problem, named *target case*, is encountered, the first step that must be taken is to retrieve a case that could be used to help solve it: the *source case.* This is known as the *retrieval* step.

Hence, if Roseline had a CBR system at her disposal, using her cookbooks as its case base, she could ask for a carrot risotto recipe and she would obtain a similar recipe, for instance a mushroom risotto recipe as

Mushroom risotto
Heat the oil and butter. Add the onion and cook until soft, about one minute. Add the rice and cook for two minutes, then add a glass of wine. Once the wine is evaporated, start adding broth, one ladleful at a time. Meanwhile, slice the mushrooms. Add them two minutes before the end.

Figure 1 – A simplified mushroom risotto recipe.

the one shown in figure 1. It turns out that there exist systems of this kind. Taaable (Cordier et al., 2013) is one of those. It uses a smooth hierarchical classification (Lieber, 2002) to find sources corresponding to a generalisation of the target: a carrot risotto is a vegetable risotto, and a mushroom risotto is a vegetable risotto too, therefore the mushroom risotto is retrieved (but not the sausage risotto).

Adaptation

The system must then figure out how the solution to the source case can be used to solve the problem. This is the *adaptation* step, also often known as the reuse step, and it will constitute the focus of this thesis. The solutions that have been proposed mostly fit into two classes:

Derivational adaptation. The reasoning process that led to building the source solution from the source problem is either stored along with the case when it is added to the case base, or otherwise induced, and is then "replayed" upon the target problem (Carbonell, 1985). This is also known as "generative adaptation" (Veloso et al., 1996).

Transformational adaptation. The target solution is constructed by starting with the source solution and modifying it with respect to the differences that exist between the source and the target problems (Carbonell, 1983).

A common solution for the adaptation task is to leave it up to the user: she is presented with the solution to the retrieved problem, and must decide whether it needs to be adapted and how. This approach is named the null adaptation and, arguably, is a type of transformational adaptation.

As for Taaable, the fact that it uses a hierarchical classification to generalise the target problem provides a first approximation of an adaptation for free: since the generalisation was made possible because carrots and mushrooms are both vegetables, then we know that one solution to the problem "How to cook a carrot risotto?" could be "Cook a mushroom risotto, using carrots instead of mushrooms." If this is the only modification made to the recipe though, the carrots will cook only two minutes, yielding a "Crunchy, undercooked carrot risotto" recipe. Therefore, the actual recipe

must be modified to take into account the effects of the ingredient substitution.

Problem Statement

Cooking problems and their solutions in the form of recipes are but a fun example of a serious class of problems. Business, manufacturing, scientific processes are described in text, often formalised as workflows, and enforced to ensure uniformity and efficiency, sometimes with the help of software applications.

Whenever the conditions that were assumed when a process was described are not present at execution time—because a chemical is missing in the manufacturing of a pharmaceutical drug, an administrative procedure was modified, or a user wants to cook a carrot risotto—the application must be able to update the process on the fly to help the user achieve her goal.

Classical planning would be one way to solve this type of problems, albeit one that requires complete domain knowledge to work. For instance, one would require precise, formalised knowledge about the physical properties of carrots before and after slicing them, and about the ideal properties that ingredients added to a risotto should have. CBR, on the other hand, can reuse existing procedures with no knowledge at all, but is still able to use any available knowledge to improve its results.

The goal of this thesis is to propose a practical model making it possible to implement a case-based reasoning system to adapt procedures where some objects are to be substituted for ones that were originally required, thereby providing Roseline with a carrot risotto recipe, and cooks around the world with recipes addressing whatever constraints suit their fancy. Because cooks are not computer scientists and training users to become fluent in a constructed language such as that of workflows is costly and unpractical, we aim at proposing a system that, to use Riesbeck and Schank's words, give "verbal answers to verbal problems".

In response to a query describing a goal in a domain-specific application, the system described herein shall be able to present the user with a consistent set of instructions enabling them to achieve that goal, expressed using natural language. For instance, when Roseline asks for a carrot risotto recipe, she could expect a recipe similar to the one shown in figure 2, in which the order in which ingredients are introduced has been altered in order to make sure that the carrots are correctly cooked.

In order to make inferences possible, though, some formal representation of the process ought to be attached to the text describing it. We propose using a temporal knowledge representation formalism for this purpose.

Achieving this goal will require exploring the interrelations between three artificial intelligence fields:

Peel and slice the carrots. Heat the oil and butter. Add the onion and cook until soft, about one minute. Add the carrots. Add the rice and cook for two minutes, then add a glass of wine. Once the wine is evaporated, start adding broth, one ladleful at a time. ~~Meanwhile, slice the mushrooms. Add them two minutes before the end.~~

Figure 2 – An adapted carrot risotto recipe—instructions added or moved are underlined, instructions removed are stroke through.

— Case-based reasoning makes it possible to use knowledge from past problems to propose hypothetical solutions to new problems, through retrieval and adaptation.
— Natural language processing (NLP) aims at providing automatic processes to turn text into a language usable by a machine (understanding), or the other way around (generation).
— Temporal knowledge representation and reasoning is interested in formalisms that make it possible to represent and manipulate knowledge about time points or intervals, and therefore also about the timing of actions.

Each of these fields is, individually, the object of a striving community, and their pairwise interfaces have received significant research interest.

The CBR community recognises that significant knowledge is available in a textual format and, consequently, that being able to exploit this text can be a great help in deploying CBR applications. Natural language is even more central in certain application domains, such as question answering, in which queries or answers, in addition to cases, may be represented as text. Therefore, the possible contributions of NLP to CBR have been a major interest from the very beginnings of CBR. This interest has been expressed, among other things, by a series of workshops on textual CBR at the 1998 AAAI conference, as well as at International and European CBR Conference 2005–2007.

The NLP community, for its part, has always had a great interest in temporal knowledge representation, and has strongly contributed to the field. Indeed, the Allen (1983) interval calculus, arguably one of the best-known temporal formalism, was designed in the first place to represent temporal information about actions described using natural language. The temporal annotation of texts, in particular, has been the subject of much work, as well as of a serious standardisation effort by Pustejovsky et al. (2003).

The analysis or the prediction of temporal processes (e.g. diagnosis) is a subject of interest in CBR and has yielded advances in temporal CBR, with time being most often expressed as sequences of events, possibly including absolute or relative timestamps. The adaptation of temporal cases, arguably,

is also an important research problem in process-oriented CBR (PO-CBR).

On the other hand, the requirements in implementing a system based on those three domains has received, to the best of our knowledge, very little attention so far. For instance, we are aware of no previous work trying to provide an integrated model for a PO-CBR system functioning with cases expressed as text and providing textual solutions as well.

Existing Solutions

A CBR system is usually evaluated empirically by its users, making it necessary to recruit domain experts. A common domain such as cooking, while seemingly dull, presents the major advantage that finding experts or quasi-experts is easy. It is therefore possible to evaluate and compare on a serious scale many different systems. This is why this domain was chosen as a common ground by the European and International Conferences for CBR to organise a competition, the *Computer Cooking Contest* (CCC) that showcases the systems developped by different teams in (originally) different application domains. The cases in all those systems are recipes, both textual and procedural in nature. It is not surprising, therefore, that most existing projects working towards similar goals as ours are also contestants in the CCC. Unfortunately, CAKE (Minor et al., 2010), arguably the most advanced system in terms of procedure adaptation has limitations on the way it handles text, whereas CookIIS (Newo et al., 2010), to our knowledge the only system—besides Taaable—that provides textual answers does no adaptation at all at the procedural level. Moreover, because of technological choices made in those two projects, we do not think it possible to simply merge them in order to provide the functionality of both.

One major effort towards adapting procedures was accomplished within the CAKE platform, an integrated process and knowledge management system. Workflows are used to represent procedural cases, more specifically "agile" workflows as defined by Weber and Wild (2005). Just like in the application we propose, those can be automatically modified on the fly in response to constraints that had not been considered during the workflow creation. For the CCC, CAKE was deployed as CookingCakeWf by Fuchs et al. (2009). It was later extended by Walter et al. (2011) and Schumacher et al. (2014) to provide automatic case acquisition from text using information extraction in the case of the former. Unfortunately, it stops short of offering a textual response. Moreover, we believe that using information extraction imposes limits both to the efficiency and to the genericity of the CookingCakeWf.

To our knowledge, the only PO-CBR system that returns a textual response is CookIIS, another contestant in the CCC. The textual adaptation step is limited in scope though. It consists in a sophisticated string replace-

ment algorithm applied on the name of the replaced ingredients. No adaptations to the process itself are applied—for instance, if the system adapts a recipe by replacing ham with bacon, it will not further adapt it by suggesting that the bacon be cooked. We think that this is a problem, and we shall expect our system, to be able to implement domain knowledge such as "bacon must be cooked before being eaten" in order to adapt the process in a consistent way.

Technical Choices

We make two technical choices that will be defended throughout this thesis as to which solution would be the most appropriate in this respect to build a system that performs well for both textual *and* procedural adaptation. Two secondary objectives inform the choices we make:

— Maximising the genericity of our solutions, in order to make them usable in application domains other than procedures—basically, in most domains involving spatial or temporal reasoning, thereby justifying the thesis's title.

— Meanwhile, maximising the amount of existing technology our solutions make use of, in order to avoid reinventing the wheel.

First, we shall argue that, while so-called "shallower" approaches inspired by information retrieval or information extraction are often preferred in textual CBR, "deep NLP" methods are more appropriate at creating the highly structured case representations, such as workflows, required for PO-CBR. Moreover, procedural texts exhibit little grammatical or structural variability—they are mostly made of sentences expressing actions to be executed, stated in the order in which they should be executed. This implies that symbolic NLP methods which sometimes yield unsatisfactory results on free text can be used to greater efficiency. Nonetheless, many benefits of using NLP are preserved, such as its greater domain portability when compared to information extraction. Finally, symbolic NLP extraction methods produce a detailed annotation of linguistic features, which is required to modify the text during adaptation.

As for the temporal knowledge representation, we shall argue in favour of using qualitative interval algebras, and more specifically the $\mathcal{INDU}$ algebra of Pujari et al. (1999), to represent processes. This implies that actions in processes are reified as time intervals. We extend this approach also to states and to durations—the latter makes it possible to avoid metric constraints, limiting the complexity of the reasoning tasks. The benefits of using an interval algebra, rather than workflows—the most commonly used representation formalism in PO-CBR—are manyfold:

— Temporal annotation in NLP generally uses qualitative algebra-inspired formalisms, making it possible to reuse existing methods and

resources.

— The possible temporal relations between events in the TimeML annotation standard (Pustejovsky et al., 2003) are inspired by the Allen calculus—and so is $\mathcal{INDU}$. This makes TimeML usable with minimal modifications, which provides us with a convenient way of keeping the text and its formal counterpart closely intermingled. As we will show, this is required in order to permit the adaptation of text.
— Merging operators have been proposed for qualitative knowledge, most notably by Condotta et al. (2008, 2009). This makes it possible to define operators for qualitative belief revision, which in turns means that the revision-based adaptation approach proposed by Lieber (2007); Cojan and Lieber (2008) can be used.
— Additionally, using qualitative algebras has the added benefit that most of the results presented in this thesis can be used in other application domains dealing with knowledge that can be expressed using those algebras, which includes many spatial or temporal applications.

Outline

Chapter 1 introduces different formalisms that can be used to build a structured case representation for processes. It also discusses the relevance of each formalism with respect to the temporal phenomena observed in processes that need to be represented. The choice of qualitative interval algebras is justified in this chapter.

Chapter 2 then describes the different options available to acquire actual case representations from procedural texts: information retrieval, information extraction, and natural language processing. Natural language processing is chosen and this choice is justified. Then, the exact methods used to acquire the representations are introduced.

Chapter 3 introduces belief revision theory and the revision-based adaptation method. It goes on to show how this method can be used to adapt a case described using a qualitative algebra. Although retrieval is not part of the subject of this thesis, it is discussed briefly, as an extension to adaptation.

Chapter 4 describes the regeneration process through which the retrieved text can be modified to correspond to the solution of the problem computed in the preceding chapter.

In chapter 5, we demonstrate that our adaptation framework has other applications besides procedural reasoning by giving an example application from the agricultural domain, in which cases are farm territories represented using a spatial algebra.

Finally, chapter 6 revisits the solutions presented in the previous chapters from the implementation point of view, describing the algorithms used

and discussing complexity issues. This implementation made it possible to compute a variety of performance metrics and conduct a large experimentation with users, presented in chapter 7.

Chapter 1

Formalisms for Procedural Case Representation

The first major CBR research area identified by Aamodt and Plaza (1994), the most influential article on CBR methodology, is that of knowledge representation. The decisions taken at this level are extremely important, because they determine how the different CBR tasks will function—most importantly in the case of this work, it determines which methods are available for case adaptation and, up to a certain point, for case acquisition.

In this chapter, qualitative interval algebra, which is the temporal knowledge representation formalism we propose to use as a case representation for procedures, will be introduced. Then, we shall survey the other formalisms that are frequently used in process-oriented CBR (PO-CBR), and especially workflows. We will briefly mention TimeML, an annotation format widely used by the natural language processing (NLP) community, to represent formally temporal information in text. Then the benefits and drawbacks of using either algebras or workflows will be weighted. Further advantages of algebras in terms of reasoning will be discussed in chapter 3. Finally, we will very quickly introduce ideas to increase the expressivity of both workflows and algebras by combining both formalisms under a common semantic framework.

1.1 Qualitative Algebras

The AI community usually divide temporal representation formalisms into two categories, depending on the type of reasoning that is intended. Reasoning on temporal constraints are usually carried on using algebraic calculi, such as interval algebras (Allen, 1981). Reasoning on actions and their effects, on the other hand, is a task well-suited to calculi descended from the situation calculus (McCarthy, 1963; Reiter, 1991).

The latter type of formalisms comes with a serious price tag: action calculi are only usable with complete domain knowledge, that is, when we are able to describe exhaustively the effect of each action on the objects present in the dynamic domain. We will not retain them, therefore, since we have defined the object of our work to be the reuse of procedural cases in the presence of possibly limited domain knowledge. Interval algebras, on the other hand, will be the main focus of this section.

The relations of a qualitative algebra define the possible ways in which two spatial or temporal entities are related to each other. Those entities can be points, intervals or regions. An algebra is generally defined based on the set $\mathfrak{B}$ of binary *base relations*, that are jointly exhaustive, i.e. they can describe all the possible relations between two entities, and pairwise disjoint, i.e. only one relation applies between two entities.

Any subset $R \subseteq \mathfrak{B}$ is interpreted as a binary relation as follows: $x\ R\ y$ if $x\ r\ y$ for some $r \in R$. Therefore, $2^{\mathfrak{B}}$ is a set of relations that is assumed to be closed under union, intersection, complement, inverse and composition: $2^{\mathfrak{B}}$ associated with those operations define a qualitative algebra.

1.1.1 Interval Algebras

Interval algebras are frequently seen types of temporal algebras. In those, the entities represent most usually closed and bounded intervals over the rational line $\mathbb{Q}$, but can sometimes be slightly different objects as will be described below.

Allen Algebra and Properties

The oldest and arguably most well-known qualitative algebra is Allen's. Allen algebra describes the 13 possible configurations between the bounds of two intervals, shown in figure 1.1. Therefore,

$$\mathfrak{B}_{\text{Allen}} = \{\mathrm{b}, \mathrm{m}, \mathrm{o}, \mathrm{s}, \mathrm{f}, \mathrm{d}, \mathrm{bi}, \mathrm{mi}, \mathrm{oi}, \mathrm{si}, \mathrm{fi}, \mathrm{di}, \mathrm{eq}\} \quad .$$

Those are the base relations of the algebra. It is also possible to express incomplete knowledge by using a subset of the base relations. This will be shown in an example immediately. Therefore, there are $2^{13} = 8192$ relations in the Allen algebra, including the impossible relation $\emptyset$, and the uninformative relation $\mathfrak{B}$, actually written $?$. By convention, a singleton relation $\{\mathrm{r}\}$ is often written r.

In figure 1.2a, the recipe text shown in figure 1 on page 2 is represented using constraints from the Allen algebra. For instance, the constraint

'heat the oil and butter' $\{\mathrm{b},\mathrm{m}\}$ 'cook the onions'

means that the interval corresponding to the heating action either precedes or meets the interval corresponding to the cooking action. A set of relations

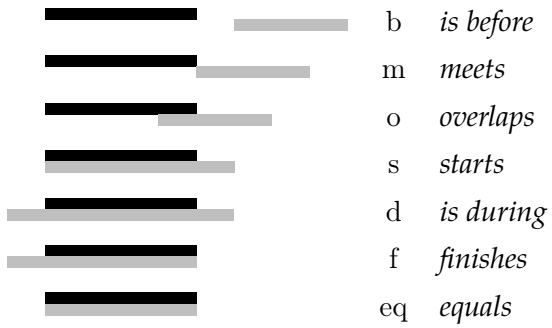


Figure 1.1 – Relations of Allen interval algebra

represents a disjunctive interpretation, i.e.

$$x \{r_1, r_2, \ldots, r_n\} y$$

for

$$r_1, r_2, \ldots, r_n \in \mathfrak{B}$$

means that either $x \{r_1\} y$, or $x \{r_2\} y$, ..., or $x \{r_n\} y$. The inverse is axiomatically distributive over the composition and over the disjunction of relations, i.e.

$$\{r_1, r_2, \ldots, r_n\}^{\smile} = \{r_1^{\smile}, r_2^{\smile}, \ldots, r_n^{\smile}\} \quad . \tag{1.1}$$

Therefore, for instance,

'heat the oil and butter' $\{b,m\}$ 'cook the onions'

could be written as

'cook the onions' $\{bi,mi\}$ 'heat the oil and butter' .

Additionally, any relation algebra is closed for the inverse, i.e. if $r \in \mathfrak{B}$, then it is certain that $r^{\smile} \in \mathfrak{B}$, and therefore, if $R \subseteq \mathfrak{B}$, then $R^{\smile} \subseteq \mathfrak{B}$.

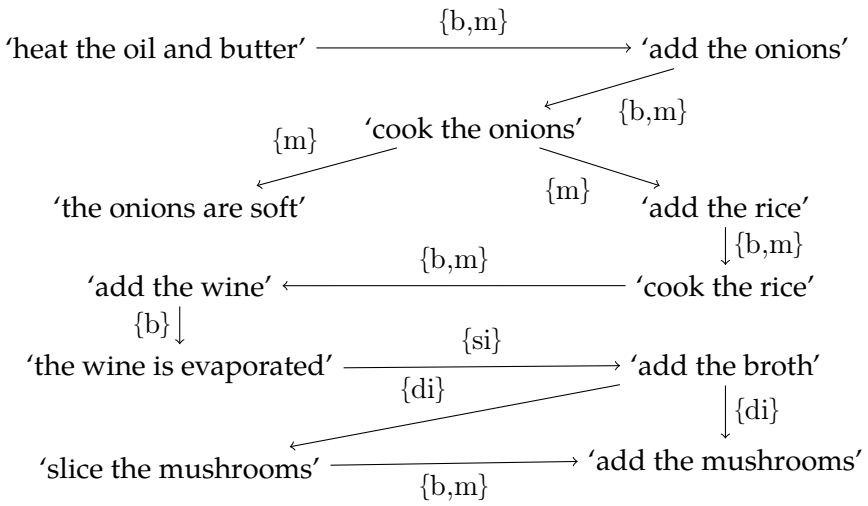
x = 'heat the oil and butter' and y = 'cook the onions' are *qualitative variables* (or variables, for short).

Other Interval Algebras

Pujari et al. (1999) proposed $\mathcal{INDU}$, a major improvement to Allen algebra. Each Allen relation can be supplemented with a duration relation, such that it is possible to express constraints on the qualitative duration of intervals. For instance, if interval x is both before and shorter than y, this can be expressed with the relation $b^{<}$. This brings the amount of possible relations between intervals to 25—not $39 = 13 \times 3$, since certain combinations, e.g. $d^{>}$, would not be satisfiable. By convention, it is allowed to write, for instance,

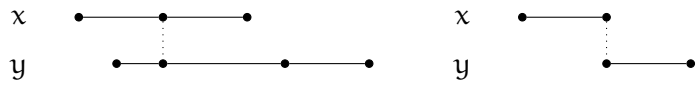
$$\left\{\begin{array}{l}
\text{'heat the oil and butter'}\ \{\mathrm{b,m}\}\ \text{'add the onions'}\\
\text{'add the onions'}\ \{\mathrm{m}\}\ \text{'cook the onions'}\\
\text{'cook the onions'}\ \{\mathrm{m}\}\ \text{'the onions are soft'}\\
\text{'cook the onions'}\ \{\mathrm{b,m}\}\ \text{'add the rice'}\\
\text{'add the rice'}\ \{\mathrm{m}\}\ \text{'cook the rice'}\\
\text{'cook the rice'}\ \{\mathrm{b,m}\}\ \text{'add the wine'}\\
\text{'add the wine'}\ \{\mathrm{b}\}\ \text{'the wine is evaporated'}\\
\text{'the wine is evaporated'}\ \{\mathrm{si}\}\ \text{'add the broth'}\\
\text{'add the broth'}\ \{\mathrm{di}\}\ \text{'slice the mushrooms'}\\
\text{'add the broth'}\ \{\mathrm{di}\}\ \text{'add the mushrooms'}\\
\text{'slice the mushrooms'}\ \{\mathrm{b,m}\}\ \text{'add the mushrooms'}
\end{array}\right\}$$

(a) Representation as a set of constraints.



(b) Representation as a graph.

Figure 1.2 – Representations of the recipe text from figure 1 in Allen algebra.



(a) Ligozat 3-4-relation $(0, 3, 4)$.

(b) Ligozat 2-2-relation $(0, 1)$, equivalent to Allen relation m.

Figure 1.3 – Examples of Ligozat's generalised interval calculi relations.

$?^{=}$ for $\{\mathrm{b}^{=}, \mathrm{m}^{=}, \mathrm{o}^{=}, \mathrm{bi}^{=}, \mathrm{mi}^{=}, \mathrm{oi}^{=}, \mathrm{eq}^{=}\}$ (i.e. all possible Allen base relations combined with $=$), as well as $\mathrm{b}^{?}$ for $\{\mathrm{b}^{<}, \mathrm{b}^{=}, \mathrm{b}^{>}\}$. $\mathcal{INDU}$ makes it possible to express knowledge such as the fact that cooking carrots take longer than cooking mushrooms:

'cook mushrooms' $?^{<}$ 'cook carrots'

Ladkin (1986) and Ligozat (1991), notably, proposed to extend interval algebra in order to make it possible to express relations between different types of "intervals". Ladkin's algebra defines a new set of relations applicable between unions of intervals—which he names "non-convex intervals". For instance, the fact that broth is added periodically to the risotto rice could be written

'cook risotto' $\{\mathrm{contains}\}$ 'add broth' ,

where 'add broth' is a non-convex interval. Ligozat's generalised interval calculi define a family of algebras for so-called "n-intervals", intervals with an arbitrary amount n of bounds. An n-interval is divided into $2n+1$ zones: $n+1$ zones are subintervals, each corresponding to an open interval between two adjacent bounds, and n zones are points, each corresponding to a bound. The zones are numbered from 0 to $2n$, making it possible to express, with respect to an n-interval x and a p-interval y, in which zone of x each bound of y is located, defining the relation between them. An example is provided in figure 1.3a. In particular, Allen algebra is a Ligozat calculus for 2-intervals. For instance, figure 1.3b shows graphically that Ligozat relation $(0, 1)$ is equivalent to Allen relation m.

Balbiani and Osmani (2000) proposed an algebra describing the possible relations between so called "c_intervals", defined over a circle rather than over a line. This makes it possible to describes additional relations. For instance, the sentence "Add milk and flour alternatively, a little at a time", in a pancake recipe, could be translated as

'add milk' $\{\mathrm{mmi}\}$ 'add flour' .

On the other hand, any notion of order is lost, so it wouldn't be possible to know whether eggs must be added before or after the milk and flour.

1.1.2 Reasoning with Qualitative Knowledge

The most relevant way of representing qualitative knowledge with respect to the current work is as a type of constraint satisfaction problems (CSP), called qualitative constraint networks (QCN) (Allen, 1983). In this context, a knowledge base is a QCN with a set of variables (in our case, representing intervals) and a set of constraints between pairs of variables. Formally, it is defined as $\mathcal{N} = (\mathcal{V}, \mathcal{C})$, where $\mathcal{V}$ is a set of qualitative variables, and $\mathcal{C}$ is a set of constraints $x\ C_{xy}\ y$ with $x, y \in \mathcal{V}$, and $C_{xy} \subseteq \mathfrak{B}$—recall that a constraint set is to be interpreted in a disjunctive way. QCNs are often visualised as directed edge-labelled graphs: $\mathcal{V}$ is the set of nodes, and $\mathcal{C}$ defines the set of arcs, i.e. each constraint defines an arc (x, y) labelled C_{xy}. Figure 1.2b on page 12 is, therefore, the visualisation of a QCN.

In QCNs, *consistency* and *satisfiability* are equivalent. A QCN is consistent if and only if it has at least one consistent scenario. A *scenario* is a QCN in which all constraints are singleton, i.e. base relations. A scenario $\mathcal{S} = (\mathcal{V}^{\mathcal{S}}, \mathcal{C}^{\mathcal{S}})$ is said to be a scenario of a QCN $\mathcal{N} = (\mathcal{V}^{\mathcal{N}}, \mathcal{C}^{\mathcal{N}})$ if all the constraints in $\mathcal{S}$ are subsets of the same constraints in $\mathcal{N}$, i.e. $\mathcal{V}^{\mathcal{S}} = \mathcal{V}^{\mathcal{N}}$ and

$$C^{\mathcal{S}}_{xy} \subseteq C^{\mathcal{N}}_{xy} \text{ for all } x, y \in \mathcal{V}^{\mathcal{S}} = \mathcal{V}^{\mathcal{N}} \quad . \tag{1.2}$$

This implies that any QCN which has $\emptyset$ as a constraint can have no scenario, and is therefore inconsistent. A *solution* of a QCN in an interval algebra is a valuation of all its variables as closed and bounded intervals in $\mathbb{Q}$ such that all the constraints defined are respected. A scenario is said to be consistent if it admits solutions. Algebras, by definition, define a partition of the set of possible variable pairs. Therefore, to any solution corresponds one and only one consistent scenario, meaning that consistent scenarios can be used as model-like objects, providing a semantics for QCNs. For this reason, the set of consistent scenarios of a QCN $\mathcal{N}$ is written $\mathcal{M}(\mathcal{N})$. For instance,

$$\mathcal{M}\left(\left\{\begin{array}{c} x\ \{b,m\}\ y \\ y\ \{f\}\ z \\ x\ \{b,o\}\ z \end{array}\right\}\right) = \left\{\left\{\begin{array}{c} x\ \{b\}\ y \\ y\ \{f\}\ z \\ x\ \{b\}\ z \end{array}\right\}, \left\{\begin{array}{c} x\ \{b\}\ y \\ y\ \{f\}\ z \\ x\ \{o\}\ z \end{array}\right\}, \left\{\begin{array}{c} x\ \{m\}\ y \\ y\ \{f\}\ z \\ x\ \{o\}\ z \end{array}\right\}\right\}$$

Qualitative algebras have a composition operation $\circ$. If, for instance

$$\left\{\begin{array}{c} x\ \{b,m\}\ y \\ y\ \{f\}\ z \end{array}\right\}$$

is known, then C_{xz} can be deduced. For the base relations, a composition table (also called transitivity table) is used. The table for Allen algebra is shown in table 1.1. The composition of disjunctive relations is the union of all possible pair-wise compositions:

$$S \circ R = \bigcup_{r \in R, s \in S} s \circ r \quad . \tag{1.3}$$

Therefore, for the above example,

$$C_{xz} = \{f\} \circ \{b,m\} = (f \circ b) \cup (f \circ m) = \{b,o,m,d,s\} \quad .$$

Qualitative algebras are closed for the composition. Therefore, if $R \subseteq \mathfrak{B}$ and $S \subseteq \mathfrak{B}$, then $S \circ R \subseteq \mathfrak{B}$.

Path consistency in a QCN is a necessary but insufficient condition for consistency. A QCN $\mathcal{N} = (\mathcal{V}, \mathcal{C})$ is path-consistent if, for every triple of constraints C_{xy}, C_{yz} and C_{xz}, the constraint C_{xz} is stronger than the constraint deduced by composition from C_{xy} and C_{yz}, i.e.:

$$C_{xz} \subseteq C_{yz} \circ C_{xy} \quad \text{for all } C_{xy}, C_{yz}, C_{xz} \in \mathcal{C} \quad .$$

For example, the QCN

$$\left\{ \begin{array}{c} x \{b,m\} y \\ y \{f\} z \\ z \{oi,mi\} x \end{array} \right\}$$

is path-consistent since, according to table 1.1 and using equations 1.1 and 1.3, we find all of the following:

$$\begin{array}{ll} \{o,m\} \subseteq \{f\} \circ \{b,m\}, & \{oi,mi\} \subseteq \{bi,mi\} \circ \{fi\}, \\ \{fi\} \subseteq \{b,m\} \circ \{oi,mi\}, & \{f\} \subseteq \{o,m\} \circ \{bi,mi\}, \\ \{bi,mi\} \subseteq \{oi,mi\} \circ \{f\}, & \{b,m\} \subseteq \{fi\} \circ \{o,m\}, \end{array}$$

Algebraic closure is the operation that consists in enforcing path consistency. For all triple of constraints C_{xy}, C_{yz} and C_{xz} in a QCN $\mathcal{N}$, all base relations that are not in $C_{yz} \circ C_{xy}$ are removed from C_{xz}:

$$C_{xz} \leftarrow C_{xz} \cap (C_{yz} \circ C_{xy}) \quad . \tag{1.4}$$

This operation is repeated until stability is attained, or until any constraint in $\mathcal{N}$ has the value $\emptyset$, meaning that $\mathcal{N}$ is inconsistent. The result of applying the algebraic closure on $\mathcal{N}$ is written $\mathcal{N}_\circ$. Path-consistency and algebraic closure are useful to optimise algorithms searching through scenarios because they allow pruning of the search space, as will be discussed in chapter 6.

1.1.3 Relation-to-Relation Distance

In a qualitative algebra, certain base relations are closer to each other than to others (Freksa, 1992). For instance, in Allen algebra, b seems intuitively closer to m than to eq. Indeed, given two intervals x and y such that $x \{b\} y$, one just has slightly to move the upper bound of x or the lower bound of y in order to obtain $x \{m\} y$. On the other hand, it would be necessary to move both bounds of either interval over a longer distance in order to obtain $x \{eq\} y$.

	b	bi	d	di	o	oi	m	mi	s	si	f	fi
b	b	?	b o m d s	b	b	b o m d s	b	b o m d s	b	b	b o m d s	b
bi	?	bi	bi oi mi d f	bi	bi oi mi d f	bi	bi oi mi d f	bi	bi oi mi d f	bi	bi	bi
d	b	bi	d	?	b o m d s	bi oi mi d f	b	bi	d	bi oi mi d f	d	b o m d s
di	b o m di fi	bi oi di mi si	o oi d s f di si fi eq	di	o di fi	oi di si	o di fi	o di si	di fi o	di	di si oi	di
o	b	bi oi di mi si	o d s	b o m di fi	b o m	o oi d s f di si fi eq	b	oi di si	o	di fi o	d s o	b o m
oi	b o m di fi	bi	oi d f	bi oi mi di si	o oi d s f di si fi eq	bi oi mi	o di fi	bi	oi d f	oi bi mi	oi	oi di si
m	b	bi oi mi di si	o d s	b	b	o d s	b	f fi eq	m	m	d s o	b
mi	b o m di fi	bi	oi d f	bi	oi d f	bi	s si eq	bi	d f oi	bi	mi	mi
s	b	bi	d	b o m di fi	b o m	oi d f	b	mi	s	s si eq	d	b m o
si	b o m di fi	bi	oi d f	di	o di fi	oi	o di fi	mi	s si eq	si	oi	di
f	b	bi	d	bi oi mi di si	o d s	bi oi mi	m	bi	d	bi oi mi	f	f fi eq
fi	b	bi oi mi di si	o d s	di	o	oi di si	m	si oi di	o	di	f fi eq	fi

Table 1.1 – Composition table from Allen (1983)

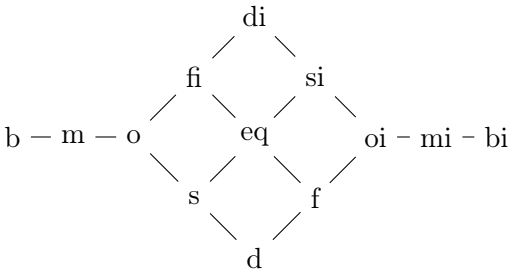


Figure 1.4 – Neighbourhood graphs for Allen algebra.

The distance between two base relations can be computed by defining what constitutes an atomic deformation on the value of a qualitative variable. If the atomic deformation on an interval x with respect to another interval y consist in either moving one bound of x which is not equal to any bound of y until it is equal to one, or conversely moving one bound of x which is equal to any bound of y until it is not equal to any, then the distance between b and m is 1—the upper bound of x is moved one step for it to be equal to the lower bound of y—whereas the distance between b and eq is 4—the upper bound of x is moved three steps so that it is successively equal to the lower bound of y, then between both bounds of y, then finally equal to the upper bound of y, and the lower bound of x is then moved one step in order to be equal to the lower bound of y.

This defines a preorder which can be translated into a graph $\mathcal{G} = (\mathfrak{B}, \mathsf{E})$, called a neighbourhood graph, in which the vertices are the base relations of the algebra, such that the distance between the vertices associated to two relations is the distance between those two relations.

For Allen algebra, using Ligozat (1991) calculus between 2-intervals offers a convenient way to obtain this preorder. Recall that a relation $\pi = (\pi_1, \pi_2)$ in this calculus is given by a pair of natural numbers $[0, 4]$. It is straightforward to compute a preorder from these relations: $\pi \leqslant \pi'$ if and only if $\pi_1 \leqslant \pi'_1$ and $\pi_2 \leqslant \pi'_2$. This yields the graph of figure 1.4.

1.1.4 Temporal Qualitative Algebras in CBR

Allen algebra has informed some work in CBR in the past. In Jaczynski (1997), cases are indexed by chronicles and temporal constraints, which are represented using a subset of Allen relations. In Jære et al. (2002), cases represent industrial incident using graphs in which certain edges are labelled with Allen relations, and the retrieval step is based on graph matching. In Montani and Portinale (2006), a temporal formalism based on points and intervals and making use of Allen relations is used to represent medical cases.

On the other hand, those systems do not exploit the full potential of

qualitative algebras. To the best of our knowledge, no system besides the one we proposed in Dufour-Lussier et al. (2012b) actually uses the inference methods defined on algebras. Moreover, what Jære et al. and Montani and Portinale propose is a domain-dependent case structure, whereas it may be possible simply to make the case equivalent to a QCN, thereby making it come within a much more generic framework.

1.2 Workflows

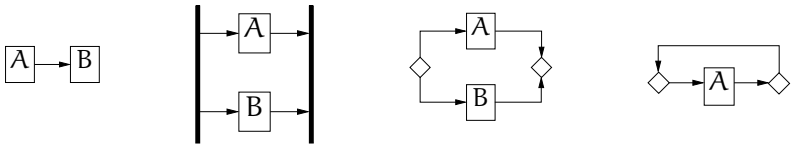
More often, in CBR, the temporal aspect is taken into account by considering sequences of events, sometimes integrating relative or absolute time stamps (Dojat et al., 1998; Ma and Knight, 2003; Sánchez-Marré et al., 2005). The most advanced work in this respect is that of process-oriented CBR (PO-CBR), in which cases are often made of activities structured using workflows.

A workflow is a formal description of a process. While workflows first appeared during the industrial revolution, workflows as we know them today were imagined during the 1980s to increase office productivity—see e.g. Ellis and Nutt (1980); Bracchi and Pernici (1984). A workflow features activities organised with a control flow. Activities are either actions accomplished autonomously by the system managing the workflow execution, or states—usually the state of the system waiting for an external trigger by an agent, or actor, that they have accomplished the required task.

The most basic control flow structure is the sequence (shown in figure 1.5a), which means that a given activity is ready to be executed as soon as another given one is done. Other control statements are the fork, yielding the concurrent execution of workflow parts, the join, synchronising forked parts, the decision, yielding to the exclusive execution of one workflow part from a set, and the merge, ending a conditional execution (van der Aalst et al., 2003). The fork and join control statements are used to create a conjunction control structure (figure 1.5b), while the decision and merge statements can be used to create either a disjunction (figure 1.5c) or a loop structure (figure 1.5d).

The workflow paradigm does not describe a language in itself. The most common way to express workflows, which is used in figure 1.5, is using UML activity diagrams (ISO/IEC 19501:2005). A more constrained syntax is useful in defining retrieval and adaptation methods, though, and many workflow languages were defined based on graphs, such as Minor et al.'s (2008), using XML in such a way as to permit knowledge-intensive retrieval (Bergmann and Gil, 2011), or the simpler approach of Kapetanakis et al. (2010), which makes it possible to deal with retrieval by using classical techniques based on subgraph isomorphism.

While a CBR system using workflows was developed by Wargitsch et al.



(a) Sequence. (b) Conjunction. (c) Disjunction. (d) Loop.

Figure 1.5 – Control flow structures.

(1997) to manage organisational memory, it was in 2005, with the proposal of agile workflows by Weber and Wild (2005), that those became a major interest for CBR research. Many applications have been developed based on agile workflows, including CAKE (Freßmann et al., 2005), a general purpose CBR system integrating a workflow engine, which is also capable of performing case-based adaptation of workflows (Minor et al., 2010), and Montani and Leonardi's (2010) system for business process monitoring, with an application in the healthcare industry.

1.3 TimeML

Many temporal formalisms have been proposed by computational linguists to annotate temporal information in text—that is, to add a formal layer inside the text itself. The most widely used is TimeML (Pustejovsky et al., 2003). This XML format makes it possible to annotate relations between actions, states, time intervals, time points, and even "non-convex intervals". TimeML has an expressivity comparable to interval algebras combining metric and qualitative constraints, but it is not associated with a semantics or any mechanism for inference. It is useful in practice, though, and will be discussed more in depth along with the implementation of our system, in chapter 6.

1.4 Adequation of Formalisms

In this section, the expressive qualities of qualitative algebras and of workflows are studied with respect to the relevant phenomena in procedures that must be formalised in order to make process-oriented reasoning possible. Consistently with Allen's theory, we consider that all actions occur over an interval of time. For convenience, actions are treated as if they were actual intervals.

By studying procedural texts, we identified few types of phenomena that structure the actions included in a procedure:

1. The *order* between the actions, which can be total or partial, including the characterisation of simultaneities.
2. The *duration* of actions, which can be described in qualitative terms using conditions based on states, or in exact or approximative quantitative terms.
3. The *repetitions* of actions, which can be executed a determinate or undeterminate amount of times, sporadically or continually, sometimes in alternation with other actions.
4. The *disjunction*, wherein an alternative course of action must be chosen depending on some conditions or on the user preferences.

As far as the order of actions is concerned, most formalisms are able to represent the basic order types, with the exception of cyclic algebras interpreted over a circle. Whereas all algebras are able to characterise simultaneity precisely—all Allen relations except b and its inverse express some type of intersection between two intervals—the only form of simultaneity that workflows can express is when two activities β and γ must not begin before a third activity α is finished nor finish after a fourth activity δ is begun.

As for durations, algebras can also express more phenomena than workflows. If states too—e.g. "the onions are caramel-coloured"—are reified as intervals, then Allen algebra can represent a qualitative duration such as that of "cook the onions until caramel-colour" as

$$\text{'cook the onions'} \ \{\mathrm{m}\} \ \text{'the onions are caramel-coloured'} \quad .$$

Workflows can do as much only by relying on an arguably unintuitive construction: a loop over a single action with an end condition corresponding to the desired state. In addition, the condition in workflow loops and disjunctions are not, to our knowledge, handled at this time by any CBR reasoning engine. If one goes further and reify actual quantitative durations as intervals, then $\mathcal{INDU}$ is able to represent both precise and imprecise durations. For instance, "cook the onions for five minutes" is

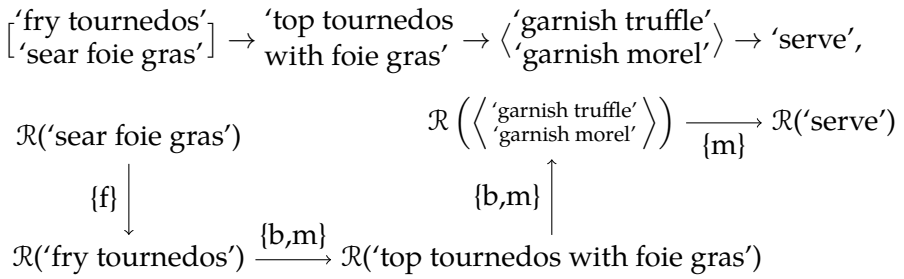
$$\text{'cook the onions'} \ \{?^{=}\} \ \text{'5 min'} \quad .$$

The repetition of actions when the number of repetitions is not known, on the other hand, is something that workflows handle natively, whereas all Allen's algebra can do is create one unique interval over which the action occurs. Ladkin's algebra could be used in those situation, but to a high computational cost.

The same goes for disjunctions of actions, which are dealt with gracefully by workflows whereas algebras can absolutely not cope except by extending the methods described in this thesis so that they work with disjunctions of QCN, making the computations more costly.

Fry the tournedos in butter. 30 seconds before the tournedos are ready, sear the foie gras. Top the tournedos with the foie gras, garnish with thin slices of truffle, and serve immediately. If you cannot find truffle, you can use morel instead.

(a) Recipe text.



(b) Associated extended workflow.

Figure 1.6 – Tournedos Rossini show the interest of combining workflow with an interval algebra.

1.5 Extension of Workflows with a Qualitative Algebra

In Dufour-Lussier et al. (2012a), we proposed defining a comparable model-theoretic semantic theory for both interval algebras and workflows, in order that both could be integrated in a common formalisms, thereby combining their benefits and eliminating their drawbacks. Figure 1.6a shows a recipe which cannot be represented entirely adequately using either formalism: the fact that the tournedos and foie gras must be ready at exactly the same time can be expressed using algebra, but not workflows, whereas the alternative that is offered between truffle and morel can be expressed with workflows, but not in any interval algebra that was introduced in this chapter. In the combined formalism we propose, the same recipe could be represented as shown in figure 1.6b

Chapter 2

Temporal Annotation of Texts

This chapter goes into the details of the case acquisition process.

The first steps are the same as in any natural language understanding system, going from the segmentation of the text into words to the identification of verb complements. Section 2.1 quickly goes through those steps. It references some commonly used tools, and especially discusses the ways in which they must be adapted to deal with issues that are specific to process description texts.

Taking the output of those first steps as input, section 2.2 proposes a way to resolve anaphoras, i.e. associate words from the text with the objects they are referring to, which is mostly specific to assembly instruction texts.

Then, using the actions, objects, and temporal clues identified by the previous steps, section 2.3 details the process through which the structured case representation itself is built.

In this chapter, a more complex recipe will be used in order to make it possible to demonstrate different linguistic characteristics occurring in procedural texts. This recipe text is shown in figure 2.1.

This work was published in Dufour-Lussier et al. (2014c).

2.1 Standard NLP Toolchain

This subsection goes through the standard steps taken by any natural language understanding system (NLU). The standard textbook for NLU is Jurafsky and Martin (2009). Several open-source tools that can greatly reduce the human cost of developing an NLU system exist. While the Stanford CoreNLP toolsuite [1] offers state-of-the-art performance for general purpose text, frameworks such as GATE [2] or OpenNLP [3] offer the best flexibility.

1. `http://nlp.stanford.edu/software/corenlp.shtml`
2. `http://gate.ac.uk/`
3. `http://opennlp.apache.org/`

Bucatini ammatriciana

— Some olive oil
— Two or three garlic cloves, crushed or (better) chopped
— One large minced onion
— One or two carrots, diced into tiny bits
— Some celery, diced as well
— A good slice of pancetta, either ground or finely chopped
— One tin of tomatoes, or the equivalent in fresh tomatoes
— 500 g pasta (bucatini work best)
— Romano cheese to taste

Sauté garlic in olive oil. Cook until brown. Discard garlic. Add the vegetables. Fry 10 minutes, stirring frequently. Stir in pancetta and cook 5 minutes. Add tomatoes and simmer 30 minutes. Meanwhile, bring water to a boil. Salt. Cook pasta until al dente. Combine pasta and sauce and sprinkle some romano. Mix well. Serve immediately.

Figure 2.1 – An example recipe, and a classic of Roman cuisine.

2.1.1 Segmentation

The very first step in processing unstructured text is to segment it into lexical items, which are more or less equivalent to words. This process, named *tokenisation*, is simple for languages which separate words with spaces, though not as simple as it may appear: for instance, "don't" is really two lexical items, while "New Jersey" is really one.

Whereas, for a language such as Chinese, tokenisation is one of the biggest NLP problems, in English, regular expressions implemented by deterministic finite-state automata are sufficient to deal with it. Ready-to-use tokenisers exist for many languages (e.g. the Stanford English Tokenizer [4]) which give quite satisfactory results.

At this stage, the text is also segmented in sentences using the punctuation as a guide. Domain specific *named entity recognition* can be performed: for instance, in the first sentence of the recipe in figure 2.1, "olive oil" corresponds to one ingredient and could thus be grouped as one lexical item and annotated with respect to an ingredient ontology.

The first sentence of the example recipe, after tokenisation, becomes: "Sauté – garlic – in – olive_oil – ."

2.1.2 Morphological Analysis

Part-of-speech tagging is the process through which each lexical item is tagged with the appropriate class. Depending on the language considered,

4. `http://nlp.stanford.edu/software/tokenizer.shtml`

different techniques are more or less appropriate. An inflectionally rich language such as Russian, in which a word can have numerous forms, is not tagged in the same way as a language such as English, in which it is for instance very usual for a related verb and noun to have the same form.

Morphologically poor languages such as English are tagged based on the likelihood of tag sequences as learnt from tagged corpora. Compare for instance the two sentences "Cream the butter" and "Butter the cream": there is no information inherent to the words "cream" and "butter" that can be used to infer whether they are being used as a verb or as a noun, but the determiner "the" makes it clear which is a noun, making the other the verb.

General purpose, ready-to-use part-of-speech taggers exist, the most well-known being the Stanford Tagger [5] (Toutanova et al., 2003), but they are not always useful when working with texts describing processes, mainly because those tend to use imperative forms, which are not so frequent in general purpose training corpora. Observe for instance the fact that verbs are unlikely to begin a sentence in normal discourse, while this is generally the case in imperative sentences. Consequently, it is preferable if the tagger can be trained on a corpus of the same type of texts that will need to be tagged.

For our implementation, we used a purpose-made annotated corpus provided by Berzak et al. (2009), consisting of 82 recipes, making 12,125 lexical items, to train a transformation-based tagger, also known as a Brill tagger. A Brill tagger first tags each lexical item with its most likely part-of-speech independently of context, then applies increasingly more specific context-sensitive rules to adjust for context Brill (1992). A typical rule likely to be induced in any corpus would be "retag any verb appearing after a determiner as a noun". The first rule induced from our corpus is "retag any noun appearing at the beginning of a sentence as a verb".

Our tagger attains 93.1% accuracy, significantly under the 97.5% state-of-the-art claimed by Søgaard Søgaard (2011), but 32% better than the same tagger trained with a general-purpose corpus.

The first sentence of the example recipe is tagged by our system as: "Sauté/verb garlic/noun in/preposition olive_oil/noun ./punctuation". [6] The Stanford Tagger trained on a general-purpose corpus mistakenly tags "Sauté" as a noun. This would cause the variable corresponding to the sautéing to be missing from the QCN.

5. `http://nlp.stanford.edu/software/tagger.shtml`

6. This is a simplification obtained by grouping together different parts-of-speech. The Penn Treebank tagset, for instance, recognises 45 parts-of-speech, including 4 for nouns and 6 for verbs.

2.1.3 Syntactic Analysis

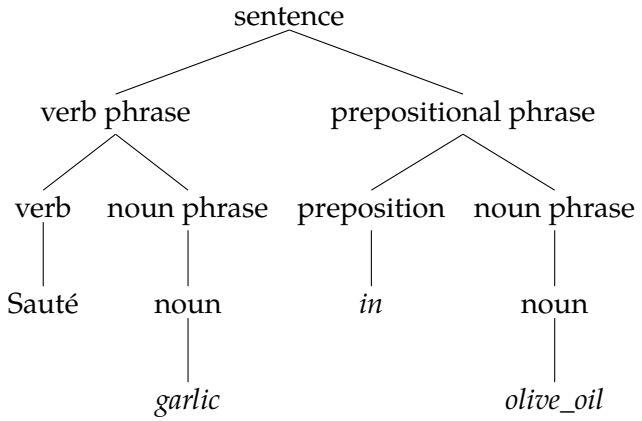
The next step is *parsing*. There are many ways to parse natural language text, depending upon the required results. Among the most effective parsers in existence today are the probabilistic context-free parsers (which give one and only one parse for any sentence) and the link parser (which gives as many parses as there are possible interpretations of the syntax of a sentence, and possibly none). A natural language parser, as a compiler, iteratively groups lexical items within phrases, until a whole sentence is grouped as one phrase, yielding a parse tree.

Parsing in TAAABLE is required only to identify the complements of verbs (which are likely to be the arguments of the actions) and the modifiers (which are likely to identify stopping conditions on the actions). Considering this, a complete parsing is not required. Indeed, since the text can be parsed clause by clause and we do not need access to such information as knowing, for instance, which noun phrase a prepositional phrase is related to, our grammar does not need recursivity.

This means that a simpler process named *chunking*, which is equivalent in complexity to a finite-state automaton, is sufficient, and a very simple grammar can be provided. For instance, here is the grammar for identifying noun phrases, formulated as pseudo-regular expressions:

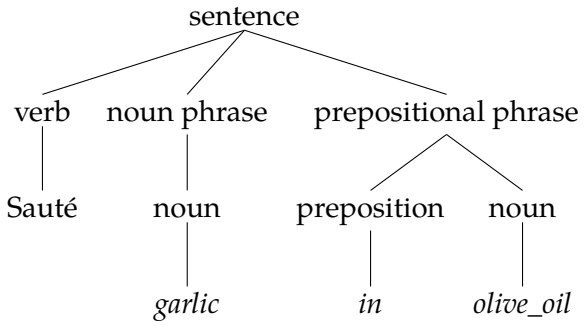
— noun = singular_noun | plural_noun
— noun_modifier = adjective | past_participle
— determiner = article | numeral | WH-determiner[7]
— noun_subphrase = pre-determiner$^?$[8] determiner$^?$ noun_modifier* noun$^+$
— noun_phrase = noun_subphrase ((comma noun_subphrase)* (comma$^?$ conjunction) noun_subphrase)$^?$

A full parser would build the following syntax tree for the first sentence in the example recipe:



7. "Which", "whose", "that", etc.
8. "All", "both", etc.

Our chunker, however, will give a simplified structure which still contains all the information required by the algorithm:



The depth of the "chunk tree" is strictly limited to 3 (root, phrases, part-of-speech tags and lexical items), whereas the depth of a regular parse tree grows with the number of lexical items.

Because procedural texts normally use simple sentence structures, the output of the chunker, the punctuation and the conjunctions are sufficient to allow for clause splitting, further dividing the text so that each finite verb [9] sits in its own group. For instance the sentence "Stir in pancetta and cook 5 minutes" is divided in two clauses by the conjunction "and": one centred around the verb "stir" and the other around the verb "cook". In language that use them, copular verbs (e.g. "to be", "to become") must be discarded in order to obtain a single clause, i.e. "until the garlic becomes brown" would be parsed as "until the garlic brown".

2.2 Anaphora Resolution

At this stage, each verb's complements and modifiers can be identified:
— Any noun phrase located after the verb and referring to available ingredients is considered as an object complement of the verb.
— Any prepositional phrase referring to available ingredients is considered as a prepositional complement of the verb.
— Any prepositional phrase referring to ingredient properties (e.g. "until brown") is considered as a relevant modifier of the verb.
— Any prepositional or noun phrase referring to a duration (e.g. "5 minutes" or "for 5 minutes") is also considered as a relevant modifier of the verb.

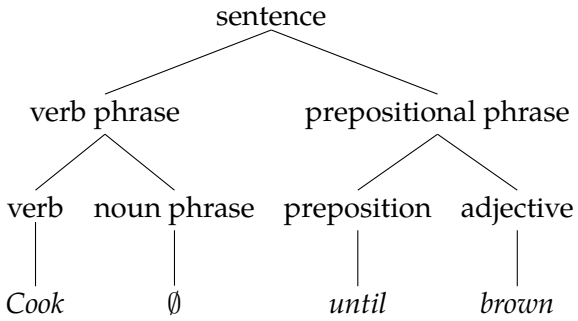
Complements are thus known at a syntactic (surface) level, but this is not always sufficient to know which actual ingredients they are referring

9. Non-finite verbs are typically those in the participle or indicative mood, e.g. "Fry 10 minutes, *stirring* continually." Finite verbs include verbs in the indicative and the imperative moods.

to. Mapping words to actual objects presents some special difficulties for instruction texts, which make heavy use of different types of *anaphoras*:

Lexical anaphoras A noun N is used to refer to some object. Instructions frequently exhibit a little-studied phenomenon known as *evolutive anaphora,* in which a word is used to refer to an object that exists at a given time, but not at another one, e.g. "Add tomatoes and simmer for 30 minutes [...] Combine pasta and *sauce*", where "sauce" references the result of the "simmer" action.

Grammatical anaphoras In common texts, those are often pronouns. In instructions though, tedious repetitions are often avoided by removing a verb's complement altogether, yielding a *zero-anaphora*, e.g. "Sauté garlic in olive oil. Cook until brown [implicitly: *the garlic*]." Zero-anaphoras are called that way because it is postulated that they exist in the sentence as pronouns having no realisation. Thus, the syntax tree of "Cook until brown" could be:



Typically, NLP systems solve anaphoras in three steps: first finding candidates referents, second filtering the list with grammatical criteria, and finally selecting one using varied heuristics. The first step can be made much easier by keeping an up-to-date set of available objects that can be passed as arguments to actions. The second step is not relevant for the present application because the anaphoras most often contain no grammatical clues about their referent. The third step is solved using a very simple heuristic. In cooking, the objects referred to by anaphoras are components: raw ingredients that are part of the set of foods created by previous actions.

2.2.1 Lexical Anaphoras

To solve evolutive anaphoras, therefore we keep a set of available food components, called *domain,* up-to-date. It is initialised with the ingredients from the ingredient list. The food components are considered as consumable resources, i.e. an action will remove some food components from the domain and add some others. For instance, the clause "mix flour, eggs and milk" would remove the three food components associated to flour, eggs and milk from the domain, and would add a new food component in it,

which becomes available to the clause "pour batter". Having an up-to-date domain makes it simpler to identify the object(s) an evolutive anaphora is referring to.

The set of food components available at step s is called $\mathcal{D}_s$, $\mathcal{D}_s \mapsto \mathcal{F}_s$ $\mathcal{D}_0$ is the initial domain, and $\mathcal{D}_{s+1}$ is the domain after the sth action in the sequential order of the text has been analysed. The domain is ordered with respect to the order of the ingredients list, which is necessary to resolve anaphoras such as "the next 5 ingredients". The order of food components added subsequently is thus not important. The objects in the domain are called "food components". A function $Ingr : F \in \mathcal{F} \mapsto I \subseteq \mathcal{I}$ is defined which, for a given food component F, gives the set $Ingr(F)$ of its ingredients.

Depending on how the domain is searched for the correct referent of an anaphora, two types are distinguished: existential and universal references.

Existential References

An existential reference is so called because it captures a food component which contains *some* specific ingredient. The ingredient is sometimes mentioned explicitly, making the reference trivial to solve, e.g. a "beef mixture" can be whatever food component contains beef. Anaphoras such as "batter" are harder to solve.

The NLP process described in the previous subsection was used to identify all the nouns in a corpus of recipes that could not be associated to a listed ingredient, most of which were indeed lexical anaphoras. Then, the set of ingredients referred to in each instance was manually built for the most frequent ones. This makes it possible to show, for instance, that a "batter" contains at least eggs or flour in over 99% of recipes from a given recipe book. This yields a target set of ingredients expected in a food component, called $ExpIngr(N)$. [10]

We consider that any food component in the domain that contains at least one ingredient in the target set is the food component referred to. In practice, there is almost always only one, but heuristics could be designed to choose one in other cases, such as taking the one with the *most* ingredients from the target set.

The set of food components that could be the referent to an existential anaphora N with target set $ExpIngr(N)$ given the domain $\mathcal{D}$:

$$Candidates(N, \mathcal{D}) = \{F \in \mathcal{D} \mid \exists i, i \in Ingr(F) \text{ and } i \in ExpIngr(N)\} \quad (2.1)$$

10. On a very large corpus, it may be possible to automate this process by setting a threshold and considering that *all* ingredients that have been mixed with others at the point where the anaphora is met are candidates for the target set.

Universal References

A universal reference is so called because it captures a set of foods for which *all* ingredients are of a certain type. This is the case, for instance, when a recipe says to "sift all dry ingredients together". Given that an ontology exists and that it is possible to map the anaphoric reference to a concept of this ontology, solving a universal reference is straightforward. For instance, "dry ingredients" represents the set of foods which are exclusively made of ingredients appearing under the "dry ingredient" concept in the ontology.

The set of food components that a universal anaphora N refers to given the domain $\mathcal{D}$, assuming a function *Class* that maps a word or an ingredient name to the corresponding concept in the ontology is:

$$\mathit{Referents}(\mathsf{N}, \mathcal{D}) = \{\mathsf{F} \in \mathcal{D} \mid \text{for all } \mathfrak{i} \in \mathit{Ingr}(\mathsf{F}), \mathit{Class}(\mathfrak{i}) \sqsubseteq \mathit{Class}(\mathsf{N})\} \quad (2.2)$$

2.2.2 Grammatical Anaphoras

The main kind of grammatical anaphora used in instruction texts is zero-anaphoras. Their resolution, while a complicated problem in general, can usually be dealt in instructions with simple heuristics. As discussed earlier, most systems need to build a list of candidate referents, though this is not necessary in this instance with the use of the domain, then filter it, which is impossible in the case of zero-anaphoras. The last step is to select one referent using preference heuristics.

The one heuristic all theories seem to agree on is recency: all other things being equal, the most recently introduced entity among the candidate referents is selected. This simple heuristic is usually sufficient in instruction texts, with a small adjustment to fit within our evolutive domain model: a zero-anaphora is a reference to the objects that were inserted in the domain by the last action that occurred before the current clause.

The most important problem they raise is their detection. A human would instinctively know that something is missing from the instruction "add milk" (i.e. "to what?"). For the computer to detect this implies a high level of linguistic knowledge. The fact that an argument is missing could be found out using either syntactic or semantic frames, the former being a much easier option to implement while the latter is less error-prone. Subcategorisation frames specify, for a given verb, the types of complements that must accompany it, e.g. "add needs an object and a prepositional complement: add O to P", making it obvious when one is missing. Subcategorisation frames can be acquired automatically in corpora, or taken from online databanks such as VerbNet (Schuler, 2005), but in any case it is advisable to verify manually that they are fit for a given application (there may be surprisingly few anyway, e.g. only 130 in our cooking application).

If the anaphora was an actual pronoun instead of being a zero-anaphora, e.g. if the second sentence of the example recipe read "Cook *it* until brown",

the result would be the same. Therefore, it is easiest to disregard the pronouns altogether, which will cause the system to think a complement is missing and use the heuristic for zero-anaphoras.

2.2.3 Updating the Domain

After resolving the anaphoras, the *actual* arguments of the action at step s are known. This subsection shows how the domain of food components available at step $s+1$ is computed from the arguments, with respect to the nature of the action.

Most actions take sets of food components as arguments. Those will be identified hereafter with capital letters, whereas sets of simple ingredients will be identified with lower case letters. The set of food components which appear as an object complement of the verb is called O, and the set of food components which appear as a prepositional complement is called P. Certain verbs may take a set of *ingredients* as an object rather than a set of food components. This is the case, for instance, of remove: "remove garlic [implicitly: from the mixture in which it is]". In this case, the set of ingredients is identifies with a lower case o.

While we consider that any food component passed as an argument to any verb is consumed by the associated action, deciding which food components are output by the action is less straightforward. Four different classes of actions are distinguished according to the food components they output. In the following, each output food component is represented as N. When there is more than one (this is defined by the action class), the set of Ns is called S.

Union actions Output only one object, the parts of which are the union of the parts of all the inputs.

$$\begin{aligned} \mathcal{D}_s = (\mathcal{D}_{s-1} \backslash (\mathsf{O} \cup \mathsf{P})) \cup \{\mathsf{N}\}, \quad \text{with} \\ \mathit{Ingr}(\mathsf{N}) = \mathit{Ingr}(\mathsf{O}) \cup \mathit{Ingr}(\mathsf{P}) \end{aligned} \tag{2.3}$$

Example: "Combine pasta with sauce":

$$\begin{aligned} \mathcal{D}_{s-1} &= \{\mathsf{F}_{\text{pasta}}, \mathsf{F}_{\text{sauce}}, \mathsf{F}_{\text{cheese}}, \ldots\} \\ \mathsf{O} &= \{\mathsf{F}_{\text{pasta}}\} \\ \mathsf{P} &= \{\mathsf{F}_{\text{sauce}}\} \\ \mathsf{N} &= \mathsf{F}_{\text{pasta\_with\_sauce}} \\ \mathcal{D}_s &= \mathcal{D}_{s-1} \backslash (\mathsf{O} \cup \mathsf{P})) \cup \{\mathsf{N}\} = \{\mathsf{F}_{\text{pasta\_with\_sauce}}, \mathsf{F}_{\text{cheese}}, \ldots\} \end{aligned}$$

Parallel actions Output as many objects as were input, implying that the action, while expressed as one clause, is executed separately for each input.

$$\mathcal{D}_s = (\mathcal{D}_{s-1} \backslash \mathsf{O}) \cup \mathsf{S} \tag{2.4}$$

where for every $F \in O$, there is exactly one $N \in S$ such that $Ingr(F) = Ingr(N)$. Example: "Mince the onions, the carrots and the celery":

$$\begin{aligned}\mathcal{D}_{s-1} &= \{F_{onion_1}, F_{carrot_1}, F_{celery_1}, \ldots\}\\ O &= \{F_{onion_1}, F_{carrot_1}, F_{celery_1}\}\\ S &= \{F_{onion_2}, F_{carrot_2}, F_{celery_2}\}\\ \mathcal{D}_s &= \{F_{onion_2}, F_{carrot_2}, F_{celery_2}, \ldots\}\end{aligned}$$

Splitting actions Input only one object and output several, e.g. "separate the egg's yolk from the white".

$$\begin{aligned}\mathcal{D}_s &= (\mathcal{D}_{s-1}\backslash\{O\}) \cup S, \quad \text{with}\\ Ingr(O) &= \bigcup_{N\in S} Ingr(N)\end{aligned} \tag{2.5}$$

Difference actions Output only one object, the parts of which are the parts of its prepositional complement input minus the part specified as its object, e.g. "remove the garlic from the oil". Difference actions are distinguished from splitting action from a practical point of view: the removed part is discarded from the domain.

$$\begin{aligned}\mathcal{D}_s =&(\mathcal{D}_{s-1}\backslash\{P\}) \cup \{N\}, \quad \text{with}\\ &Ingr(N) = Ingr(P)\backslash\{o\}\end{aligned} \tag{2.6}$$

2.3 Case Acquisition

Because instruction texts are usually designed such that the instructions are to be applied in the textual order of their description, building an $\mathcal{INDU}$ QCN from them is not very complicated. Once all the actions described in the text have been fully analysed and the verb modifiers are known, generating a QCN primarily consists in applying a mapping from the set of modifiers to the set of $\mathcal{INDU}$ relations.

Two actions textually juxtaposed without further precision are considered as taking place sequentially. Therefore, disregarding the "until brown" condition for now, the first two sentences of the example recipe of figure 2.1 are represented as:

'sauté(garlic,olive_oil)' $\{\mathrm{b,m}\}^?$ 'cook(garlic)' $\{\mathrm{b,m}\}^?$ 'discard(garlic)' .

Frequently, the duration of an action is given qualitatively by specifying a condition under which the action can be stopped, as in "Cook pasta until al dente." It is expected for the condition to appear in a prepositional complement, where it can be extracted. This can be represented by considering

that "al dente" is a state of "pasta" which is reified as an interval, as was hinted to in section 1.4. Because the action finishes immediately when the state is attained, the relation $\mathrm{m}^?$ is used:

'cook(pasta)' $\{\mathrm{m}\}^?$ 'al_dente(pasta)'

The other type of duration information that can appear in verb modifiers is explicit, quantitative duration, such as in "Fry 10 minutes". This can be represented by reifying the time, creating an interval whose sole reason for existing is to last 10 minutes, and using the $?^=$ relation:

'fry(vegetables)' $?^=$ '10 min'

$\mathcal{INDU}$ can also be used to represent simultaneity This can be triggered by verb modifiers such as "meanwhile", but also simply by the use of the participle mood, e.g. "Fry 10 minutes, stirring constantly." The latter case yields a simultaneity that is scoped only over the previous action. Therefore, it makes sense to represent this kind of simultaneity as inclusion using the relation $\mathrm{d}^?$:

'stir(vegetables)' $\{\mathrm{d}\}^?$ 'cook(vegetables)'

"Meanwhile"-like modifiers are more tricky: they initiate a sequence of actions that can be simultaneous with the preceding action. In this case, we assume that the simultaneity extends over a sequence ending just before an action which uses as input any food component which is an output of any action in the sequence. For instance, the simultaneity sequence initiated by "Meanwhile, bring water to a boil" goes on until just before "Combine pasta and sauce", because the combining action takes as input the pasta, which is output by the cooking action located inside the sequence. It is sufficient to use the $\mathcal{INDU}$ relation $\{\mathrm{d,oi,f}\}^?$ between the first action following the simultaneity sequence and the first action of this sequence—the other relations can be inferred

2.4 Porting to a Different Domain

This section gives an overview of the additional work that would be required to make the method described above work in a different domain. An experimental protocol is provided in figure 2.2 to serve as an example.

The standard NLP toolchain described in section 2.1 can be applied in exactly the same way to this experimental protocol as to recipes. This very example shows some sentences that are not in the imperative, but since this also occurs in some recipes, the part-of-speech tagger trained on recipes gives the expected results on the protocol.

1. Suspend evenly 1 kg. of yeast in 4 liters of 0.5 M Na_2HPO_4 and boil for 3 hours.
2. Cool to 37° and add 0.5 gm. of trypsin (Wilson 1:250) on the 1st, 3rd, 6th, and 10th day of incubation at 37°. Add 5 ml. of toluene on the 1st day, and 2.5 ml. of toluene on the 3rd, 7th, 11th, and 15th day of incubation. Incubate for 16 days at 37°, with occasional stirring, and adjust pH daily to 7.8–8.0 by the careful addition of NaOH.
3. Collect precipitate by centrifugation. Discard supernatant.

[...]

7. The precipitate from step 6 is added rapidly with vigorous stirring:–
   (a) to 8 liters of *absolute ethyl alcohol*; stir for 1 hour; centrifuge.
   (b) The precipitate is added to 4 liters of *absolute ethyl alcohol* and treated as in (step a).
   (c) The residue is dried *in vacuo.*
   (d) The dried powder is refluxed for 3 hours with 500 ml. of *absolute ethyl alcohol.*
   (e) Centrifuge and dry immediately *in vacuo* and store *in vacuo.*

Figure 2.2 – Excerpts from a scientific experimental protocol for preparing zymosan (Pillemer et al., 1956).

For the anaphora resolution, though, domain-specific resources are required: the ontology of objects, the dictionary of actions, and the target sets of lexical anaphoras.

The ontology must contain the chemicals that may be used in the protocol, such as yeast, trypsin or Na_2HPO_4, and is used to find those objects in the text and to resolve lexical anaphoras. The dictionary of actions must contain the relevant verbs, such as "suspend", "boil" or "centrifuge", along with their subcategorisation frame to allow identification of zero-anaphoras, and their class for domain updating. For instance, "suspend" is a union action requiring both an object and a prepositional complement, and "centrifuge" is a splitting action requiring but an object.

Regarding lexical anaphoras needing target sets to resolve, it is less clear what may be the requirements of different application domains. The semi-automatic process used for building the target sets in the cooking domain rely on the ingredient list, but it may also be used in texts without lists, provided that the words used as anaphora do not appear as lexical variants in the ontology. Otherwise, those anaphoras may be difficult to deal with. An additional piece of information about lexical anaphoras which is required for scientific protocols is the names through which the outputs of a splitting action can be referred as. In cooking, the only obvious case in which a splitting action assigns a specific name to its outputs is the separation of an egg into its yolk and white. In scientific protocols, most splitting

actions assign names to their outputs irrespective of the objects considered: for instance, the outputs of "centrifuge" in the example protocol are "precipitate" or "residue" and "supernatant". Therefore the application must know that, e.g. in step 3, "precipitate" refers to one of the two outputs of the "centrifuge" action, and "supernatant" to the other.

Some sentences are in the passive voice and have subjects instead of objects. The subjects are not considered when looking for action arguments. This has no practical effect in this case, as the missing argument is interpreted as a zero-anaphora and correctly resolved. Larger scale testing of the algorithm in domains other than cooking would make it possible to say whether this can be considered as a valid heuristic or whether this is a stroke of luck. In the latter case, it would be straightforward to analyse the sentence correctly, since the output of the part-of-speech tagger makes it possible to infer whether a verb is in the passive voice.

As far as the actual case acquisition step is concerned, what is suggested is mostly sufficient to deal with the protocol shown. Because the rules were built by observing the phenomena in recipe texts, some additional rules would be required for other domains. For instance, all the durations and temporal relations are correctly identified by the algorithm, except for the parts of step 2 in which the order of the text does not respect the temporal order. In the expression "treated as in (step a)", if one can identify the "step a" and it refers to a series of actions in which each takes as input the output of the previous, as is actually the case, it is easy to copy those actions. Identifying the step by its number would require the implementation to remember the said number at the clause segmentation stage—the current implementation internally numbers the clauses with no respect to any numbering that may be present.

Chapter 3

Reasoning with Temporal Cases

The two previous chapters have justified the choice of case representation formalism we used and explained how those case representations can be acquired automatically from text. The current chapter will show how those case representations are used to propose solutions to new problems.

Whereas chapter 1 has attempted justifying our choice of qualitative algebras over workflows from a knowledge representation point of view, we will see that using algebras also offers very interesting trade-offs in terms of reasoning.

This chapter first introduces belief revision theory, which is required to tackle revision-based adaptation, in section 3.1, both in general and as specifically applies to QCNs.

This revision framework, introduced in section 3.2, is well-suited to handle problems expressed in terms of substitutions in QCNs. In the mushroom risotto recipe, all instances of actions effected upon mushrooms will be replaced with corresponding actions on carrots. Because we assume all relevant domain knowledge to be present as well, this is likely to cause the QCN to become inconsistent—we know, for instance, that carrots are not done after two minutes cooking. Intuitively, belief-based adaptation offers a way of repairing the result of the substitution, making the smallest possible amount of changes to it—cooking the carrots a bit longer, for instance, inverting the "add rice" and "add carrots" actions in the process—such that consistency with the domain knowledge is restored.

In practice, revision of QCNs under substitution is not that easy, because the postulates of belief revision theory (and the revision algorithms, as we will see in chapter 6) require consistency to be preserved at all times. This forces us to formulate the problem in a more complex way, which is detailed in section 3.3.

Finally, although we chose to talk mainly of adaptation, we hint in sec-

tion 3.4 at how the notions introduced in this chapter to serve the needs of CBR adaptation can also be relevant to retrieval, through the adaptation-guided retrieval paradigm.

This work was published in Dufour-Lussier et al. (2012b).

3.1 Belief Revision

Belief revision is a logical field of study which is concerned with the minimal modifications of beliefs about a static world. From an epistemic point of view, we consider an agent that holds, at some point in time, some beliefs about the world. Say, the year is 1483, and young Nicolaus believes that the Earth is at the centre of the world, that the celestial spheres are moving around it, and that Polish sausage is seasoned with garlic. Let us call those beliefs ψ. At some further point in time, the agent learns new beliefs, which he holds as definitely true, but that contradict partially his former beliefs. Say, grown up Nicolaus has found out that, as a matter of fact, the Sun is at the centre of the world. Let us call those new beliefs μ.

The object of study of belief revision is to determine how the beliefs of Nicolaus will be modified by this discovery. Certainly he cannot hold it true that the Earth is at the centre of the world, since that would contradict outright his new beliefs. Meanwhile, he has no reason to change his beliefs about the celestial spheres, except insofar as they could not possibly be moving after all. As for Polish sausage, there is no obvious reason why their ingredients should be changed. The beliefs ψ of the agent after having been minimally modified by some new beliefs μ are written $\psi \dotplus \mu$. This reads "ψ revised by μ".

Formally, in a given knowledge representation formalism, a revision operator $\dotplus$ maps two belief bases ψ and μ to a belief base $\psi\prime \wedge \mu$, where $\psi\prime$ is the minimal modification of ψ such that the conjunction of $\psi\prime$ and μ is consistent. If the conjunction of ψ and μ is in fact consistent, i.e. the new beliefs do not contradict the old ones, then the revision operator should return this conjunction,

A contraction operator can also be defined. From an epistemic point of view, belief contraction occurs when an agent has doubt about beliefs he previously held as true. A contraction operator $\dot{-}$ maps two belief bases ψ and μ, to $\psi\prime$, which is the minimal modification of ψ such that $\psi\prime$ does not entail μ.

There is a relation between revision and contraction, which was formalised by Gärdenfors (1988, 1992) as the Harper identify after Harper (1976)

$$\psi \dot{-} \mu = \psi \vee (\psi \dotplus \neg\mu) \tag{3.1}$$

and the Levi identity after Levi (1977)

$$\psi \dot{+} \mu = (\psi \dot{-} \neg\mu) \wedge \mu \quad . \tag{3.2}$$

3.1.1 Axiomatic Belief Revision

The necessity for a formal model of belief revision in epistemic agents was first suggested by Levi (1977). A thinking agent X is considered, along with her corpus $K_{X,t}$ of knowledge at time t. Suppose that, at this time, X knows that some proposition h holds, i.e. $K_{X,t} \models h$. Levi wants to determine how X will change his corpus if, at time t', $t' > t$, she learns that h does not hold any more: $K_{X,t'} \models \neg h$. It understood that any knowledge g at time t that is not falsified by $\neg h$ may be useful and should be preserved at time t'. Therefore, the retraction of h should cause the smallest possible change to $K_{X,t}$.

Alchourrón, Gärdenfors, and Makinson (1985) created an influential model for belief revision, which got to be known as the "AGM model", and which proposes a set of axioms that any belief revision operator $\dot{+}$ should confirm. Those axioms were reformulated for propositional logic by Katsuno and Mendelzon (1991) as follows. Given the belief bases ψ, ψ_1, ψ_2, μ, μ_1, μ_2, and φ:

Postulate 1. $\psi \dot{+} \mu \models \mu$.

This is called the success axiom. The definition of belief revision implies that the new beliefs have precedence over the old beliefs. The success axiom, enforces this by making sure that the result of the revision is consistent with the new beliefs.

Postulate 2. *If* $\psi \wedge \mu$ *is consistent, then* $\psi \dot{+} \mu \equiv \psi \wedge \mu$.

This is called the vacuity axiom. If the new beliefs can actually be added to the old beliefs without causing them to become unconsistent, then they should simply be added.

Postulate 3. *If* μ *is consistent, then* $\psi \dot{+} \mu$ *is consistent.*

This is called the consistency axiom. If the new beliefs are satisfiable, the result of the revision is satisfiable.

Postulate 4. *If* $\psi_1 \equiv \psi_2$ *and* $\mu_1 \equiv \mu_2$, *then* $\psi_1 \dot{+} \mu_1 \equiv \psi_2 \dot{+} \mu_2$.

This is called the extensionality axiom. It enforces the principle of irrelevance of syntax: if two belief bases are equivalent, the result of their revision by two equivalent belief bases should be equivalent.

Postulate 5. $(\psi \dot{+} \mu) \wedge \varphi \models \psi \dot{+} (\mu \wedge \varphi)$.

Postulate 6. *If* $(\psi \dot{+} \mu) \wedge \varphi$ *is consistent, then* $\psi \dot{+} (\mu \wedge \varphi) \models (\psi \dot{+} \mu) \wedge \varphi$.

Those are respectively called the superexpansion axiom and the subexpansion axiom. They enforce the minimality of change.

3.1.2 Distance-Based Revision

Dalal (1988) proposed a revision operator which computes the revision of two propositional belief bases ψ by μ by exploring the models of μ and retaining only those that are closest to ψ:

$$\mathcal{M}(\psi \dot{+} \mu) = \{m \in \mathcal{M}(\mu) \mid \min_{p \in \mathcal{M}(\psi)} d(p, m) = \Delta\} \quad , \tag{3.3}$$

where d is a distance function applicable to the models of ψ and μ—Dalal uses the Hamming distance applied to propositional logic—and

$$\Delta = \min_{p \in \mathcal{M}(\psi), m \in \mathcal{M}(\mu)} d(p, m) \quad .$$

Using this definition, it is possible to create a distance-based revision operator for any language that respects those three constraints:

1. It has a model theory.
2. The set of all models is finite.
3. There exists a distance function applicable between the models of formulas in this language.

Condotta et al. (2008, 2009) have shown that this is the case for the language of QCNs, and has created a belief merging operator for this language.

3.1.3 Revision of Temporal Beliefs

As seen in chapter 1, qualitative knowledge bases can be formulated in terms of constraint satisfaction problems (QCNs). Additionally, any consistent QCN has a set of scenarios, which can serve as models of the QCN. This is useful because the set of scenarii of a QCN is necessarilly finite—and, indeed, not larger than $|\mathfrak{B}|^{|\mathcal{C}|}$ for a QCN $\mathcal{N} = (\mathcal{V}, \mathcal{C})$ over an algebra defined by $\mathfrak{B}$—whereas the set of solutions of a consistent QCN over an infinite domain is necessarily infinite.

The other requirement to be able to derive a distance-based revision operator is to have a distance function applicable between those scenarii. The relation neighbourhood graphs for qualitative algebras introduced in subsection 1.1.3 can be used to this end. While more sophisticated ways of computing the semantic distance between two scenarii have been proposed recently by D'Almeida et al. (2012), we will use a fairly simple function in this work, in which the total distance between two scenarii with an equal set of variables is the sum of the distances between their sets of atomic constraints. That is, for two scenarii $\mathcal{S}^1 = (\mathcal{V}, \mathcal{C}^1)$ and $\mathcal{S}^2 = (\mathcal{V}, \mathcal{C}^2)$ the distance d between them is:

$$d = \sum_{x,y \in \mathcal{V}} d(C^1_{xy}, C^2_{xy}) \tag{3.4}$$

From this point, the revision of temporal belief $\psi \dot{+} \mu$ is reduced to a search within the model space of μ for those models minimising the distance to any model of ψ. As in any model-based revision operator, the result of the search is a set of models, i.e. scenarii. Unlike in propositional logic, though, Schwind (2010) has proven that, given a revision of two QCNs $\psi \dot{+} \mu$, there might not exist one QCN $\mathcal{N}$ such that $\mathcal{M}(\mathcal{N}) = \mathcal{M}(\psi \dot{+} \mu)$. Therefore, the result of the revision of two QCNs is not a QCN itself, but a set of scenarii.

If the two QCNs have a different set of variables, the solution we propose is adding all missing variables from ψ into μ and vice-versa. All constraints can be initialised using the $?$ relation, then, given that the intersection of the set of variables of ψ and μ is not empty, strengthened by applying the algebraic closure.

Revision in the Propositional Closure of a Qualitative Algebra

Although we reckon that a revision operator on QCNs giving as result a set of scenarii is satisfactory for our application, we propose a temporal revision method on an extension of qualitative algebras, such that the result is necessarily representable in the formalism (Dufour-Lussier et al., 2014a,b).

The idea is that, given two QCNs under disjunctive normal form,

$$\Psi \equiv \bigvee_i \psi_i \quad , \qquad M \equiv \bigvee_j \mu_j \quad ,$$

the distance between them is the minimum of the distance between their disjuncts,

$$d(\Psi, M) = \min_{i,j} d(\psi_i, \mu_j) \quad . \tag{3.5}$$

Given that the disjuncts are QCNs with constraints expressed in regular qualitative algebras, we can then define a revision operator for QCNs in qualitative algebras closed under disjunction based on the existing revision operator. The latter is modified so that, instead of returning a set of scenarii, it returns a disjunction of scenarii. The result of the revision is therefore the disjunction of the results of the pairwise revision of the disjuncts which are minimally distant:

$$\Psi \dot{+} M \equiv \bigvee_{d(\psi_i, \mu_j) = d(\Psi, M)} \psi_i \dot{+} \mu_j \quad . \tag{3.6}$$

Being able to process revision with disjunction makes it possible to process revision with negation, by applying the De Morgan laws until all negations are atomic. The negation of an atomic constraints is defined as the complement of the relation, making it possible finally to eliminate atomic negations:

$$\neg(x \; R \; y) \equiv x \; \mathfrak{B} \backslash \; R \; y \quad . \tag{3.7}$$

Besides making it possible to obtain the result of the revision expressed in the same formalism as its operands, this approach has many advantages:

— It makes it possible to express certain constraints more naturally, and using less variables, which reduces computation time. For instance, given two variables x and y such that $x \{b\} y$, and given a third variable z which is equal to x or is equal to y, it is possible to write

$$z \{eq\} x \vee z \{eq\} y \quad .$$

Without disjunction, it would be necessary to introduce an additional variable, for instance a variable i_{xy} representing the interval between the end of x and the beginning of y:

$$x \{m\} i_{xy} \wedge i_{xy} \{m\} y \quad .$$

Only then would it be possible to express that z is either equal to x or after it, equal to y or before it, and disjoint from but adjacent to the interval between x and y:

$$z \{eq,bi\} x \wedge z \{eq,b\} y \wedge z \{m,mi\} i_{xy}$$

— It makes it possible to express procedural disjunction as defined in section 1.4, wherein different sequences of actions must be accomplished depending on certain conditions, thus bridging a part of the gap between qualitative algebras and workflows.

— By using the Harper identity it is possible to define a belief contraction operator from the existing belief revision operator.

3.2 Revision-Based Adaptation

Intuitively, belief revision and CBR adaptation are similar processes. After the user requested a recipe for a carrot risotto the system was able to retrieve a mushroom risotto recipe. The target is the request of a risotto with carrots. The source is the request of a risotto with mushrooms, and the solution to the source is a mushroom risotto recipe. The solution to the target should be a carrot risotto recipe. One way to make a transformational adaptation would be to retain as much as possible from the source solution, only changing whatever must be changed in order for the solution to satisfy the target. That is, mushrooms will be "retracted" from the solution, and carrots will be added in their place. If domain knowledge about vegetable cooking is included, a mere substitution of mushrooms by carrots will give an inconsistent results, and further modifications of the solution will be required. The use of a belief revision operator guarantees that the solution suggested will be consistent (consistency axiom), and that it will indeed satisfy the

target (success axiom) while remaining as close as possible to the source solution (superexpansion and subexpansion axioms).

Revision-based adaptation was first proposed by Lieber (2007). The intuition behind this framework is that the best way of reusing case solutions is by first applying the retrieved solution to a target problem as is, adding whatever constraints are mandated by the target problem and the relevant domain knowledge. If the source is not as a matter of fact reusable as-is, this conjunction will be inconsistent, and it will become necessary to repair it. The repaired conjunction is therefore the solution to the target problem.

As much of the source as can be made consistent must be retained, hence the name "conservative adaptation" which was originally given to this theory. Making a minimal change to some knowledge to make it consistent with newer knowledge is exactly what belief revision is about, so it seems only logical that conservative adaptation theory be formalised under the auspices of belief revision theory.

Therefore, given a revision operator $\dot{+}$, the adaptation of the source solution `Source` to the target `Target` with respect to some domain knowledge `DK` can be expressed simply by

$$\texttt{Source} \wedge \texttt{DK} \dot{+} \texttt{Target} \wedge \texttt{DK} \quad .$$

3.3 Formulation of the Adaptation Problem as a Revision Problem

Let us consider again the risotto adaptation problem. Because the QCN representing this entire recipe is large, only the constraints on vegetable and rice cooking will be shown. If `Source` is a QCN $(\mathcal{V}_{\texttt{Source}}, \mathcal{C}_{\texttt{Source}})$, we shall write that

$$\mathrm{C}_{\texttt{Source}} = \left\{ \begin{array}{l} \text{'cook(mushroom)'} \; ?^{=} \; \text{'2 min'} \\ \text{'cook(mushroom)'} \; \{\mathrm{f}^{<}\} \; \text{'cook(rice)'} \end{array} \right\} \quad .$$

The first constraint represents the fact that the mushrooms are cooked for two minutes, and the second constraint the fact that they are added so as to be ready at the same time as the rice. The constraint between 'cook(rice)' and '2 min' is not represented because it is unspecified in the text. By applying algebraic closure, it can be deduced to be $?^{>}$.

The relevant domain knowledge includes the cooking durations, the order between the durations, and the fact that when the action of cooking

something is finished, that thing is cooked. For $\mathtt{DK} = (\mathcal{V}_{\mathtt{DK}}, \mathcal{C}_{\mathtt{DK}})$:

$$C_{\mathtt{DK}} = \left\{ \begin{array}{l} \text{'cook(carrot)'} \stackrel{?}{=} \text{'25 min'} \\ \text{'cook(mushroom)'} \stackrel{?}{=} \text{'2 min'} \\ \text{'cook(rice)'} \stackrel{?}{=} \text{'18 min'} \\ \text{'2 min'} \stackrel{?}{<} \text{'18 min'} \\ \text{'2 min'} \stackrel{?}{<} \text{'25 min'} \\ \text{'18 min'} \stackrel{?}{<} \text{'25 min'} \\ \text{'cook(carrot)'} \{\mathrm{m}\}^{?} \text{'cooked(carrot)'} \\ \text{'cook(mushroom)'} \{\mathrm{m}\}^{?} \text{'cooked(mushroom)'} \\ \text{'cook(rice)'} \{\mathrm{m}\}^{?} \text{'cooked(rice)'} \end{array} \right\} \quad .$$

As for the target, it does not contain any constraint that can be easily represented using $\mathcal{INDU}$, therefore, for $\mathtt{Target} = (\mathcal{V}_{\mathtt{Target}}, \mathcal{C}_{\mathtt{Target}})$, we shall write $\mathcal{C}_{\mathtt{Target}} = \emptyset$.

On the other hand, there remains to be integrated into the revision the subsitution of mushrooms by carrots, which will be written "mushroom $\rightsquigarrow$ carrot". If the substitution is applied directly in `Source`, the conjunction of substituted-`Source` and `DK` will include all three of 'cook(carrot)' $\stackrel{?}{=}$ '2 min', 'cook(carrot)' $\stackrel{?}{=}$ '25 min' and '2 min' $\stackrel{?}{<}$ '18 min', which is inconsistent.

In a model-based revision $\psi \dot{+} \mu$, if ψ is inconsistent, i.e. it has no model, it cannot be used to guide the exploration of the models of μ. Therefore, the result will be μ:

$$\text{If } \mathcal{M}(\psi) = \emptyset, \text{then } \mathcal{M}(\psi \dot{+} \mu) = \mathcal{M}(\mu) \quad . \tag{3.8}$$

This means discarding the source solution altogether and building a target solution from scratch using only the domain knowledge, which defeats the purpose of CBR.

The solution we propose involves making the substitution a two-step problem: first an abstraction is applied in the left term of the substitution, such that "mushroom" becomes an abstract variable x, then a refinement is applied in the right term, such that all intervals related to x are equated to intervals related to "carrot".

Parametrised QCNs.

It is assumed that the variables of the considered QCNs representing actions and states can be parametrised. Parameters can be concrete or abstract. Concrete parameters denote concept of the application domain, e.g. "mushroom" in the interval 'cooking(mushroom)'. The set of parameters is $\mathcal{P}$. A parameter $p \in \mathcal{P}$ is either a *concrete parameter*, $p \in \mathcal{CP}$, or an *abstract parameter*, $p \in \mathcal{AP}$, i.e. $\mathcal{P} = \mathcal{CP} \cup \mathcal{AP}$, but not both, i.e. $\mathcal{CP} \cap \mathcal{AP} = \emptyset$.

QCN Conjunction

Let $\mathcal{N}_1 = (\mathcal{V}^1, \mathcal{C}^1)$ and $\mathcal{N}_2 = (\mathcal{V}^2, \mathcal{C}^2)$ be two QCNs. Their conjunction, noted $\mathcal{N}_1 \wedge \mathcal{N}_2$, is the QCN $\mathcal{N} = (\mathcal{V}, \mathcal{C})$ such that $\mathcal{V} = \mathcal{V}^1 \cup \mathcal{V}^2$ and $\mathcal{C}$ contains the constraints of $\mathcal{C}^1$ and $\mathcal{C}^2$, such that, if C^1_{xy} and C^2_{xy} are both defined for some x, y, C_{xy} retain the most restrictive constraint, i.e. $C_{xy} = C^1_{xy} \cap C^2_{xy}$.

Substitutions.

The *atomic substitution* $\sigma = p \rightsquigarrow q$, where $p, q \in \mathcal{P}$, is the function from $\mathcal{P}$ to $\mathcal{P}$ defined by

$$\sigma(a) = \begin{cases} q & \text{if } a = p \\ a & \text{otherwise} \end{cases} \quad . \tag{3.9}$$

A *substitution* is a composition $\sigma_1 \,;\, \ldots \,;\, \sigma_n$ of atomic substitutions σ_i. The composition of σ and σ', denoted by $\sigma \,;\, \sigma'$, is the function that associates to $p \in \mathcal{P}$

$$\sigma \,;\, \sigma' \,(p) = \sigma'(\sigma(p)) \in \mathcal{P} \quad . \tag{3.10}$$

Let $\sigma = p \rightsquigarrow q$ be an atomic substitution. σ is *concrete* if p and q are both concrete. σ is an *atomic abstraction* if p is concrete and q is abstract. σ is an *atomic refinement* if p is abstract and q is concrete. A *concrete substitution* is a composition of concrete atomic substitutions. An abstraction is a composition of atomic abstractions, and a refinement is a composition of atomic refinements.

Any concrete substitution σ can be written $\sigma = \alpha \,;\, \rho$ where α is an abstraction and ρ is a refinement, as the following equation illustrates:

$$\texttt{mushroom} \rightsquigarrow \texttt{carrot} = \texttt{mushroom} \rightsquigarrow x \,;\, x \rightsquigarrow \texttt{carrot}$$

where $\texttt{mushroom}, \texttt{carrot} \in \mathcal{CP}$ and $x \in \mathcal{AP}$. This can be shown as follows. First, σ can be written $p_1 \rightsquigarrow q_1 \,;\, \ldots \,;\, p_n \rightsquigarrow q_n$ with $p_i, q_i \in \mathcal{CP}$ and $p_i \neq p_j$ if $i \neq j$. Let $x_1, \ldots, x_n$ be n abstract parameters, let $\alpha_i = p_i \rightsquigarrow x_i$, let $\rho_i = x_i \rightsquigarrow q_i$, let $\alpha = \alpha_1 \,;\, \ldots \,;\, \alpha_n$, and let $\rho = \rho_1 \,;\, \ldots \,;\, \rho_n$. α is an abstraction, ρ is a refinement and $\sigma = \alpha \,;\, \rho$.

Let σ be a substitution. σ is extended on qualitative variables by applying it to their parameters. For example, if $\sigma = \texttt{mushroom} \rightsquigarrow \texttt{carrot}$ then $\sigma(\,\texttt{cooking}\,(\,\texttt{mushroom}\,)) = \texttt{cooking}\,(\,\texttt{carrot}\,)$. Then, σ is extended to a constraint C_{xy} by

$$\sigma(x \; C_{xy} \; y) = \sigma(x) \; C_{xy} \; \sigma(y) \quad . \tag{3.11}$$

Finally, σ is extended on a QCN by applying it to its variables and constraints:

$$\sigma((\mathcal{V}, \mathcal{C})) = (\sigma(\mathcal{V}), \sigma(\mathcal{C})) \tag{3.12}$$

where

$$\sigma(\mathcal{V}) = \{\sigma(x) \mid x \in \mathcal{V}\} \tag{3.13}$$
$$\sigma(\mathcal{C}) = \{\sigma(C) \mid C \in \mathcal{C}\} \quad . \tag{3.14}$$

Adaptation problem.

An adaptation problem is given by a tuple $(\texttt{Source}, \texttt{Target}, \texttt{DK}, \sigma)$. Source and Target are the representations of the source and target cases by QCNs with concrete variables, i.e. not parametrised by any abstract parameter. DK is a QCN representing the domain knowledge. $\sigma = p_1 \rightsquigarrow q_1 \; ; \ldots ; p_n \rightsquigarrow q_n$ is a concrete substitution such that each p_i parametrises a variable of Source, and each q_i parametrises a variable of Target. $\texttt{DK} \wedge \texttt{Source}$ must be consistent in order to obtain a meaningful adaptation, and $\texttt{DK} \wedge \texttt{Target}$ must be consistent in order to obtain an adaptation at all. They are assumed to be. The goal of adaptation is to build a consistent QCN AdaptedCase that entails $\texttt{DK} \wedge \texttt{Target}$, whose qualitative variables are obtained by applying σ onto the qualitative variables of Source, and that is obtained through a minimal modification of $\texttt{DK} \wedge \texttt{Source}$.

3.3.1 Principles of revision-based adaptation of a QCN

A first idea to perform the adaptation, given $(\texttt{Source}, \texttt{Target}, \texttt{DK}, \sigma)$, is to apply σ on Source, thus obtaining a QCN $\texttt{DK} \wedge \sigma(\texttt{Source})$ that may be inconsistent, and then restoring consistency. Although this gives a good intuition of the revision-based adaptation of a QCN, it is not consistent with the irrelevance of syntax principle. Indeed, any two inconsistent knowledge bases (e.g. two inconsistent QCNs) are equivalent: their sets of models are both empty. Thus, at a semantic level, repairing an inconsistent knowledge base is meaningless. By contrast, revision aims at modifying a *consistent* knowledge base with another *consistent* one, the conjunction of which may be inconsistent.

The revision-based adaptation consists first in decomposing σ in an abstraction α and a refinement ρ: $\sigma = \alpha \; ; \rho$—cf. previous section. Then, α is applied to Source: a QCN $\texttt{DK} \wedge \alpha(\texttt{Source})$ is built that is necessarily consistent since $\texttt{DK} \wedge \texttt{Source}$ is consistent and every constraint of $\texttt{DK} \wedge \alpha(\texttt{Source})$ corresponds to a constraint of $\texttt{DK} \wedge \texttt{Source}$. In other words, $\texttt{DK} \wedge \texttt{Source}$ is consistent and is more or equally constrained as $\texttt{DK} \wedge \alpha(\texttt{Source})$, so $\texttt{DK} \wedge \alpha(\texttt{Source})$ is consistent.

The third step involves revision. The idea is to make a revision

$$\texttt{DK} \wedge \alpha(\texttt{Source}) \dot{+} \texttt{DK} \wedge \texttt{Target} \wedge \mathcal{N}_\rho \tag{3.15}$$

where $\mathcal{N}_\rho$ represents the following statement: "Each qualitative variable x of $\alpha(\texttt{Source})$ is constrained to be equal to its refinement $\rho(x)$." For this purpose, the EQ relation is used (in $\mathcal{INDU}$, this is $\mathrm{eq}^=$): x EQ $\rho(x)$. Therefore,

$\mathcal{N}_\rho = (\mathcal{V}_\rho, \mathcal{C}_\rho)$ where

$$\mathcal{V}_\rho = \alpha(\mathcal{V}) \cup \sigma(\mathcal{V}) \tag{3.16}$$

$$\mathcal{C}_\rho = \{x \text{ EQ } \rho(x) \mid V_i \in \alpha(\mathcal{V})\} \tag{3.17}$$

μ is consistent since $\mathtt{DK} \wedge \mathtt{Target}$ is and since each constraint x EQ $\rho(x)$ of $\mathcal{N}_\rho$ either is an empty QCN (when x does not contain any abstract parameter refined by ρ) or links a variable x that does not appear in $\mathtt{DK} \wedge \mathtt{Target}$ with $\rho(x)$.

Then, $\psi \dot{+} \mu$ gives a set of scenarios and `AdaptedCase` is chosen among them.

3.4 Applications to Retrieval

As stated in the introduction, the application we developed rely on the TAAABLE system (Cordier et al., 2013) to perform the retrieval and suggest substitutions. This section discusses the possibility of using the method proposed for adaptation to provide retrieval as well as a possibility for future work.

Smyth and Keane (1998) argue in favour of adaptation-guided retrieval: rather than retrieving the case corresponding to the most similar problem, a CBR system should attempt to retrieve the case whose solution is the easiest to adapt into a solution to the target problem.

Thanks to this retrieval model, revision-based adaptation of QCNs as defined in this chapter can actually provide a framework for retrieval. We have stated that, given ψ built from the source and μ built from the target, we can find a distance δ between ψ and μ such that all interpretations of μ that are at distance δ from any interpretation of ψ are candidate solutions to the target.

A first step towards adaptation-guided retrieval of QCNs would be to find the distance δ_{CaseBase} between μ and all possible ψ built upon all the cases in the case base, i.e. between $\mathtt{DK} \wedge \bigvee_i \mathtt{Source}_i$ and $\mathtt{DK} \wedge \mathtt{Target}$. Given this, all the models of μ that are at distance δ from any interpretation of any ψ are candidate solutions to the target.

This does not, however, take into consideration the substitution problem. If the substitution is not suggested by the retrieval process of TAAABLE, it will need to be integrated in the scenario searching process. A solution would be, given any concrete parameters `p` found in `Target` but not in `Source`, to attempt matching it against all concrete parameters `q` found in `Source` but not in `Target`, and retain the mapping that minimises the distance.

The computational cost of this would be formidable, however. Solutions exist that would make it possible to diminish the cost in exchange for sacrificing correctness. For instance, Bergmann and Stromer (2013) propose a

two-step retrieval process for workflows, in which the cases that seem less relevant are filtered out with an approximative algorithm before a more costly case selection algorithm is applied to the remainder of the case base. A similar solution could be retained to filter less likely source candidates, for instance based on the maximum distance between QCNs, which can be computed in polynomial time—computing the exact distance is exponential.

Additionally, an ontology of foods—which exists already in Taaable—could be used to filter out less likely substitution candidates.

Chapter 4

Adaptation of Textual Cases

One major originality of this thesis is the proposal to apply the case-based reasoning adaptation process on the textual description of the solution. This would typically involve natural language generation, a costly process that typically produces text of lower quality than human-written text.

An alternative, which can be used when a source text is available—for instance in text summarisation or simplification, or in paraphrasing—is *language reuse and regeneration* (Radev, 1999). Of course, in textual case-based reasoning, a source text is available by definition.

Language reuse consists in finding a sentence that has the desired semantic representation from a corpus and using it directly in the target text. We used this approach in Dufour-Lussier et al. (2010): when an ingredient *a* is to be replaced with an ingredient *b* in a retrieved recipe, the sequence of culinary actions applied to *a* in the text are identified and replaced with the part of another recipe text which describes the closest sequence of culinary actions possibly applied to *b*.

Language regeneration goes a step further, as it allows the copied text to be modified in order to make it fit better or augment its quality.

In this chapter, we sketch out a proposal to apply a language regeneration process to the retrieved recipe text in parallel to the revision of its formal temporal representation. We first describe the preconditions—that is certain specificities in how temporal annotation and temporal reasoning must be carried—in section 4.1. The general solution we propose is presented in section 4.2. Finally, in section 4.3, we present some related work.

Although this chapter presents preliminary research, the ideas presented therein have been implemented. This is described in the implementation chapter, in section 6.3.

4.1 Preconditions

The usual approach for using textual cases for process-oriented case-based reasoning is to extract case representations, such as a workflows or a graphs, then deal with these representations without regard to their textual origin.

A system that would aim to answer a user's request with an adapted text, though, would need to maintain cross-references between the extracted representations and the original texts. In this way, whenever the representation would be modified by the adaptation process, an equivalent modification could be introduced in the text, such that the text–to–representation mapping is maintained.

An efficient way to maintain the cross-references between the variables in our qualitative constraint networks and the words and clauses that realise them in the text is to use a temporal annotation language.

TimeML (Pustejovsky et al., 2003) seems to be the most expressive such language in existence. It makes it possible to represent, directly in text, using XML markup, the relations between actions, states, intervals, time points, and even sets of time points. Figure 4.1 shows one possible representation of the instructions "Two minutes before the end, add the mushrooms" in TimeML—with certain uninformative attributes removed.

Tasks in a process are represented as `EVENT`s. `TIMEX3` tags encode explicit temporal expressions that are related to event and specify the time of their occurrence or their duration, whereas `SIGNAL` tags underline words or phrases that are used to indicate the temporal relations between events. Those relations are annotated using the `TLINK` tag with a `relType` attribute that offers values comparable to the base relations of interval algebras—it distinguishes two types of inclusion (and their inverse) and two types of equality, but does not cover overlap, for a total of 14 relations.

To make textual adaptation possible, it may be necessary to move or delete sentences or sentence parts. Therefore, more annotations are needed. Everything that may become a candidate for movement or deletion—linguistics teaches those can only be phrases, as identified by the chunker described in subsection 2.1.3—must be marked. Clauses, even though they are identified before chunking, are also phrases, and candidates for movement or deletion, and thus must be marked.

An important consideration in text regeneration is anaphoric cohesion. For instance, in a sentence such as "Two minutes before the end, add the mushrooms", there is a zero-anaphora acting as a prepositional complement to the verb "add": the mushrooms are added *to* something, say to the rice mixture. If this sentence is moved, then the text is re-analysed, the anaphora should still be identified, and should be resolved either to the same thing, or to something acceptable. How to do this—and what is

```
1  <TIMEX3 tid="t3" type="duration-before-next" value="PT2M">
2    Two minutes
3    <SIGNAL sid="s4">
4      before
5    </SIGNAL>
6    the end
7  </TIMEX3>
8  ,
9  <EVENT eid="e12" class="OCCURENCE">
10   add
11 <EVENT>
12 the mushrooms .
13 <MAKEINSTANCE eiid="ei12" eventID="e12" tense="INFINITIVE" pos="VERB"/>
14 <TLINK eventInstanceIS="ei12" relatedToEvent="ei11" relType="AFTER"/>
15 <TLINK eventInstanceIS="ei12" signalID="s4" relatedToTime="t3" relType="BEGINS"/>
```

Figure 4.1 – A recipe excerpt annotated with simplified TimeML.

acceptable—is discussed further below, and requires that the result of the anaphora resolution must be memorised.

Another addition is what we name the "empty signal". As explained in section 2.3, we consider that two actions that are sequential in the text, without further precision, are implicitely sequential in time. This assumption is materialised as an empty `SIGNAL` tag, which can be referenced in `TLINK`s and which can, if dictated by the adaptation, be replaced with a different, explicit signal.

Lastly, while the TimeML `relType` attribute is included and filled as well as possible, in order to make our application more easily reusable, it is not actually used: the possible values for this attribute do not form a relation algebra, and are therefore not relevant for the types of reasoning we use. Therefore, all temporal relations are also specified in $\mathcal{INDU}$, or in Allen—if a relation R is specified in Allen algebra, the equivalent $\mathcal{INDU}$ relation $\mathrm{R}^?$ is to be read.

Figure 4.2 shows the sentence "Two minutes before the end, add the mushrooms" annotated using TimeML enriched with the information just described and more information that are useful for various tasks. Line 2 shows an empty signal. Lines 3–16 shows that the `TIMEX3` element constitutes a noun phrase, and therefore could be wholly moved or removed if needed, without making the sentence grammatically incorrect. Finally, lines 32–34 shows how the zero-anaphora was resolved. Again, certain uninformative attributes are not shown.

The framework that will be described in the next section relies on being able to access the deductive closure of the temporal relations in a text. Computing the closure of a set of temporal annotations has been studied before with the aim of comparing and improving the quality of annotations. Setzer et al. (2003) have been the first ones to tackle this issue, but implemented a solution that only covers five of the 14 TimeML relation types.

```
<cp:clause>
  <timeml:SIGNAL sid="s13" cp:impliedSignal="true" cp:value="implicit-after"/>
  <cp:NP>
    <timeml:TIMEX3 tid="t3" type="duration-before-next" value="PT2M"
     cp:virtualInterval="2min" cp:supportInterval="support_t3(2min)"
     cp:approximate="false">
      <cp:token id="tok84" pos="CD">Two</cp:token>
      <cp:token id="tok85" pos="NNS">minutes</cp:token>
      <timeml:SIGNAL sid="s4" cp:impliedSignal="false"
       cp:value="starts-support-interval">
        <cp:token id="tok86" pos="CS">before</cp:token>
      </timeml:SIGNAL>
      <cp:token id="tok87" pos="AT">the</cp:token>
      <cp:token id="tok88" pos="NN">end</cp:token>
    </timeml:TIMEX3>
  </cp:NP>
  <cp:token id="tok89" pos=",">,</cp:token>
  <cp:VP>
    <timeml:EVENT eid="e12" class="OCCURENCE" cp:input="f_mushroom_1 f_mixture_4"
     cp:output="f_mixture_5" cp:ingredients="i0 i1 i2 i3 i4 i5"
     cp:actionClasses="add gather" cp:interval="e12=add(mushroom)">
      <cp:token id="tok90" pos="VB">add</cp:token>
    </timeml:EVENT>
  </cp:VP>
  <cp:NP>
    <cp:token id="tok91" pos="AT">the</cp:token>
    <cp:ingredient ingref="i5" foodref="f_mushroom_1">
      <cp:token id="tok92" pos="NNS">mushrooms</cp:token>
    </cp:ingredient>
  </cp:NP>
  <cp:token id="tok93" pos=".">.</cp:token>
  <cp:ingredient anaphora="true" anaphoraType="empty"
   syntacticFunction="prepositional" ingref="i0 i1 i2 i3 i4"
   foodref="mixture_4"/>
  <timeml:MAKEINSTANCE eiid="ei12" eventID="e12" tense="INFINITIVE" pos="VERB"/>
  <timeml:TLINK eventInstanceIS="ei12" relatedToEvent="ei11" signalID="s13"
   relType="AFTER" cp:allenRel="{bi,mi}"/>
  <timeml:TLINK eventInstanceIS="ei12" signalID="s4" relatedToTime="t3"
   relType="BEGINS" cp:allenRel="{s,eq}"/>
  <cp:VirtualSupportRelation timeml:relatedToTime="t3" cp:induRel="{?=}"
   cp:induConstr="support_t3(2min){?=}2min"/>
</cp:clause>
```

Figure 4.2 – A recipe excerpt annotated with simplified CraqpotML.

Verhagen (2005) has proposed a more thorough solution. His solution owes much to the path-consistency algorithms used to close qualitative constraint networks, but has the particularity of mapping the resulting relations onto the TimeML relation types. Its use would be advantageous if one did not want to or could not extend TimeML.

In the case of our application, this would offer no real benefit. Instead, we use regular algebraic closure as defined in subsection 1.1.2, enforced using Vilain and Kautz's (1986) path-consistency algorithm, described in subsection 6.2.2

4.2 General Principles

This section describes the general principles of the text regeneration algorithm we propose. The process was designed with revision-based adaptation in mind, but is in most parts general enough so that it could be applied to any situation in which a temporal relation between two events needs to be changed, on the request of an external agent or process, without having to provide a justification. One mandatory condition is that each change should be propagated through all the temporal relations, such that algebraic closure is maintained.

Another, non-trivial condition is that a temporal knowledge extraction process be available. This process must be symbolic, at least insofar as there exists a mapping from annotated linguistic features onto sets of qualitative constraints. The reason for this constraint is that, in the spirit of Kay (1975), Appelt (1987), Shieber (1988) and others that worked on reversible, or bidirectional grammars—used for both syntax analysis and generation—we explore the notion that the rules that are used to extract temporal information from a text can also be exploited to modify the same text.

Let us name this linguistic features–to–temporal relations mapping

$$\textit{Annotate} : \text{set of linguistic features} \mapsto \mathcal{INDU} \text{ constraint} \quad . \tag{4.1}$$

This mapping is defined thanks to a set of annotation rules. The textual adaptation framework will now be presented, in a first time with respect to an idealised, bijective version of *Annotate*. We will then discuss the characteristics that must be added to handle first a mapping that is not injective, then a mapping that is not surjective. Finally, we will explain how anaphoras are dealt with.

4.2.1 Adapting Text With a Bijective Annotation Mapping

If we assume *for the moment* that a bijection exists between relevant sets of linguistic features and $2^{\mathfrak{B}}$, the set of relations in the algebra used, text re-

generation with respect to a modified temporal relation between two events is a "simple" process of

1. Finding the set of linguistic features that maps to R, given the new constraint $x\ R\ y$, where x and y are the intervals corresponding to the occurence of two events.
2. Modifying in the least costly way possible the corresponding features of the utterances that realise the two events so that they correspond to those that map to R.

In our application, the linguistic features used are:

— The part-of-speech of the word realising the events. For instance, in "Add the onion and fry on low fire until soft", there are three events: "add" and "fry" are occurence events realised by verbs, while "soft" is a state event realised by an adjective.
— In the case of verbs, their mood and tense. In the above example: present imperative for both verbs.
— Additionally, in the case of verbs, their modifiers, including `SIGNAL`s and `TIMEX`es. In the above example: an empty signal for both verbs, and "until soft" for "fry".
— In the case of adjectives, the subordinating conjunction that commands the adjective. In the above example: "until" for "soft".
— The membership of events in clauses, and of clauses in sentences. In the above example, "add" is part of one clause, "fry" and "soft" are parts of a different clause,[11] and both clauses and no other clause form a sentence.
— The textual order of clauses and sentences.

Because sentence structures in procedural texts are not usually rich, clause in sentence membership as described above suffices. A general purpose application would need to take subordination into account.

Simple Example

Using those features, here are two examples of rules defined by the annotation process described in chapter 2:

Annotation rule 1. *Given:*
— e_1 *and* e_2 *are culinary events*
— e_1 *and* e_2 *are realised by verbs is the same tense and mood*
— e_2 *has a modifier which is an empty or "then" signal*
— *the clauses of* e_1 *and* e_2 *are in the same sentence and the former immediately precedes the latter* or *the clause of* e_1 *is the last clause of sentence* s_1 *and the clause of* e_2 *is the first clause of sentence* s_2 *and* s_1 *immediately precedes* s_2

11. In a more general-purpose application, "until soft" may be analysed as belonging to a different clause, since it means "until the onion is soft". In procedural texts, it makes sense to ignore copular verbs such as "to be", and therefore not create an additional clause.

Annotate: 'e_1' $\{b,m\}^?$ 'e_2'

Annotation rule 2. *Given:*
- — e_1 *and* e_2 *are culinary events*
- — e_1 *is realised by a present imperative verb*
- — e_2 *is realised by a present participle verb*
- — *the clauses of* e_1 *and* e_2 *are in the same sentence*
- — *there is no other clause in this sentence* or *the clause of* e_2 *immediately follows the clause of* e_1
- — *there is no signal specifying a relation between* e_1 *and* e_2

Annotate: 'e_1' $\{di\}^?$ 'e_2'

Consider the sentence "When the rice is done, remove from the stove, add the butter and the parmesan, mix thoroughly, cover, and let rest for five minutes." There are six events, realised by "done", "remove", "add", "mix", "cover", and "let rest", and five clauses—"done" and "remove" are in the same clause, as explained above. The first event "done" is a state and is in a relation with "remove" such that the lower bound of their intervals are the same, i.e. $\{s,eq,si\}^?$. All other events have an empty signal and are in a $\{b,m\}^?$ relation with their respective preceding event, with respect to the rule 1.

As a simple example, if a user or an external process requests that 'add' $\{b,m\}^?$ 'mix' become 'add' $\{di\}^?$ 'mix', rule 2 is identified as applying. It is therefore necessary to make minimal linguistic modifications such that the new text corresponds to the first element of this rule. The first, second and fourth criteria are met. The second option of the fifth criterion is met, therefore, applying the principle of the minimality of change, no change is needed. The last criterion requires deleting the empty signal, but does not change the form of the sentence. Finally, meeting the third means changing the verb mood and tense, and therefore its inflection—specifically, it requires adding the suffix "-ing" to the verb "mix".

Therefore, the modified form of the sentence is "When the rice is done, remove from the stove, add the butter and the parmesan, mixing thoroughly, cover, and let rest for five minutes."

While very simple, this example shows one great benefit of using text regeneration: every linguistic element that is not directly relevant to the temporal reasoning task, such as the adverb "thoroughly" in this case, does not need to be analysed correctly or indeed at all, and will be preserved nonetheless.

Clause movement

As a slightly more complex example, consider the case in which a user or an external process requests that 'add' $\{b\}^?$ 'cover'—which is the result of algebraic closure by application of $\{b,m\}^? \circ \{b,m\}^? = \{b\}^?$—become 'add' $\{di\}^?$

'cover'.[12] In common with the preceding example, the mood, tense, and therefore inflection of "cover" must be changed, and the empty signal must be removed. On the other hand, no option of the fifth citerion is met. In order to obey this criterion, one clause must be moved next to the other.

If we postulate that only one clause will move, there are four possibilities: either the clause of e_1 moves immediately before or after the clause of e_2, or the opposite. Different strategies could be used to determine which clause should be moved. We have identified two, that make use of the algebraic closure.

The first is to decide, for each possible movement, whether the resulting text would be consistent with the algebraic closure of the constraints. After changing 'add' $\{\mathrm{b}\}^?$ 'cover' to 'add' $\{\mathrm{di}\}^?$ 'cover' one relevant constraint can be deduced: 'cover' $\{\mathrm{b}\}^?$ 'mix'. On the other hand, 'add' $\{\mathrm{b,m}\}^?$ 'mix' is maintained. This implies that moving the "add" clause and not moving the "cover" clause would lead to at least two inconsistencies. Therefore, the "cover" clause must be moved.

Moreover, the minimality of change principle could be used to suggest that "cover" be moved *after* "add": otherwise, new sentences will need to be created. Therefore, the modified form of the sentence is "When the rice is done, remove from the stove, add the butter and the parmesan, covering, mixing thoroughly, and let rest for five minutes."

The second strategy involves postulating the existence of a "natural" order of events in text. This order would coincide with the order on the lower bound of the intervals over which the events occur.

This involves dividing the Allen algebra base relations in three categories:

— $\mathsf{B} = \{\mathrm{b}, \mathrm{m}, \mathrm{o}, \mathrm{di}, \mathrm{fi}\}$
— $\mathsf{E} = \{\mathrm{eq}, \mathrm{s}, \mathrm{si}\}$
— $\mathsf{A} = \{\mathrm{bi}, \mathrm{mi}, \mathrm{oi}, \mathrm{d}, \mathrm{f}\}$

therewith defining a total preorder $\leqslant$ on intervals. Given $x\ R\ y$:

— If $r \in \mathsf{B}$ for all $r \in R$, then $x < y$.
— If $r \in \mathsf{A}$ for all $r \in R$, then $x > y$.
— In all other cases, $x \sim y$—i.e. $x \leqslant y$ and $x \geqslant y$.

This preorder is applied to the relations between the intervals over which the events occur to build a strict total preorder on them. For the example above, this is:

$$\text{"done"} \sim \text{"remove"} < \text{"add"} < \text{"cover"} < \text{"mix"} < \text{"let rest"}$$

Then, in order to minimise the modifications applied to the text, we ensure that all relations are inequalities, by discriminating with the actual textual

12. This change does not make much sense, but this is not for the textual adaptation task to judge.

order. For the example above, this gives:

"done" < "remove" < "add" < "cover" < "mix" < "let rest"

This order shows that the "cover" clause should be moved to immediately after the "add" clause, which is the same result as the one obtained with the first strategy.

While we have not proven it, we think that applying those strategies has the implication that the text modification maintains consistency with respect to the algebraic closure of the associated qualitative constraint network.

4.2.2 Taking Into Account the Non-Injectiveness of the Annotation

The actual *Annotate* mapping used by the annotator may be non-injective. For instance, those different utterances ought to be annotated in the same way:

— "Add the butter. Mix thoroughly."

— "Add the butter. Then, mix thoroughly."

— "Mix thoroughly, after having added the butter."

Indeed, the very rules shown as example above had a disjunctive first element, and probably should have been formalised non-injectively as four different sets of linguistic features mapping to two different qualitative constraints.

This amounts to selecting among paraphrases, which is a common task in natural language generation systems. Different formalisms force different approaches, but systems generally have a module which is responsible for choosing a realisation based on syntactic, pragmatic or stylistic constraints (Iordanskaja et al., 1991; Matthiessen, 1991; Bateman, 1997).

Regeneration offers an alternative, that can be derived from the strategy we suggested above when the left element is disjunctive, which is to minimise the changes. This minimises the adaptation effort and should, therefore, minimises the risk of introducing mistakes. A proper implementation of this strategy would make it necessary to define a distance measure on the linguistic feature sets, which is beyond the scope of this thesis.

4.2.3 Taking Into Account the Non-Surjectiveness of the Annotation

Just like the *Annotate* mapping may be non-injective, it is not expected to be surjective. While natural language is certainly capable of generating utterances for the 2^{25} $\mathcal{JNDU}$ relations by using disjunctions, it does not appear desirable to define rules such that, for instance, "Before or after adding the butter, mix thoroughly" be mapped to 'add' $\{\mathrm{b,bi}\}^?$ 'mix', or that "Add the butter and, while doing so, mix thoroughly, going on mixing until after the butter is added" be mapped to 'add' $\{\mathrm{oi}\}^?$ 'mix'.

We take the standpoint that it is also not desirable to generate this type of sentence, although it could be argued that the validity of this choice is domain-dependent. The implication is that, when the user or the external process requests that a constraint be changed to a relation that does not constitute the right side of any rule, a strategy must be used to determine which rule to use in its stead.

Given a requested change from $\mathrm{x}\ \mathrm{R}\ \mathrm{y}$ to $\mathrm{x}\ \mathrm{S}\ \mathrm{y}$, we propose to apply this strategy: select a S' that constitutes the right side of a rule and that has those properties, in this order:

1. It minimises the relation-to-relation distance to S, as defined subsection 1.1.3.
2. It minimises the relation-to-relation distance to R.
3. It is the largest, i.e. has the most relations.

and revise the qualitative constraint network by $\mathrm{x}\ \mathrm{S}'\ \mathrm{y}$. Criterion 1 enforces maximum integrity with the change request, while criteria 2 and 3 diminish the effect of propagation as much as possible.

This strategy has the following desirable effects in case the intersection or R and S is not empty:

— If $\mathrm{R}\subseteq\mathrm{S}$, do not make any change. This eliminates the risk of mistake, and ensures consistency between the text and the qualitative constraint networks since a constraint relaxation will not have propagated during algebraic closure.

— Otherwise, if there exists a relation S' that constitutes the right side of some rule, such that $\mathrm{S}'\subseteq\mathrm{R}\cap\mathrm{S}$, the strategy guarantees that the selected new relation will be one of those S'. This corresponds to a constraint strengthening and may cause strengthening in other constraints when it propagates, but will avoid other types of constraint change.

4.2.4 Anaphoric Cohesion

As shown above, attention must be paid to anaphoras in order to maintain what Siddharthan (2006) names "anaphoric cohesion". An additional process must be performed, after a clause has been moved or deleted, to ensure that anaphoras can still be resolved in a satisfactory way.

If one defines, for now, "satisfactory" as meaning that an anaphora in the regenerated text should refer to the exact same object as the same anaphora did in the original text, then the process is straightforward.

For instance, consider this cooking instruction: "Cut the onions, then peel the potatoes and boil them. Combine." The user may learn that potatoes are easier to peel once boiled, and request that the text be changed such that

$$\text{'boil(potato)'}\ \{\mathrm{b,m}\}^{?}\ \text{'peel(potato)'}\quad .$$

If the text adaptation process is applied as described up to now, the result will be "Cut the onions, then boil them and peel the potatoes", which is incorrect because "them" now looks like it refers to the onions.

Given that the anaphora "them" was resolved correctly as refering to the same thing as "the potatoes" before regeneration, it can simply be replaced with its antecedant: "Cut the onions, then boil the potatoes and peel the potatoes." This sentence is correct but stylistically leaves something to be desired: an anaphora should be generated to remove the second occurence of the phrase "the potatoes". There has been much work on the generation of referring expressions which has explored this problem, including some from the point of view of procedural text generation (Tutin and Kittredge, 1992; Kosseim et al., 1996) and even some from the point of view of regeneration (Siddharthan, 2006). We do not think that we could offer a better solution that what exists already.

What is more complicated is defining what a satisfactory resolution should be exactly in the context of procedural texts. Consider the following excerpt from a non-simplified version of a risotto recipe:

> [...] Add the rice and gently warm the mixture until the rice becomes transparent. Raise the temperature and pour the wine in. Let evaporate, stirring constantly. Then add the stock, one laddleful at a time, while the rice cooks–about 18 minutes. Two minutes before the end, add the mushrooms. [...]

The last sentence corresponds to what is shown in figure 4.2, in which a zero-anaphora is shown as being resolved to the mixture of vegetable stock, olive oil, onion, rice and white wine, named `mixture_4`, which exists at the specific time when the "add mushrooms" action begins.

If, as would happen in the running example, this action is moved to before the "add rice" action, the anaphora will resolve to the mixture of olive oil and onion, named `mixture_1`, which exists at the specific time when the "add rice" action used to begin. If this is considered not satisfactory, the anaphora will be explicited, which is problematic because at this time, neither `mixture_4` nor any equivalent mixture exists.

We therefore propose an extension of the definition of a satisfactory anaphora resolution, which is specific to procedural texts: an anaphora is resolved in a satisfactory way if it is resolved either as referring to the same object, or as referring to an earlier or later stage of the same object's transformation. This concept is defined with respect to a directed acyclic graph tracing object transformations, e.g. the one shown in figure 4.3 for the risotto recipe. In this example graph, the referent of the zero-anaphora is marked with ◈. The earlier stages, marked with ● are the ones from which it is possible to reach ◈. The later stages, marked with ◇, are the ones that can be reached from ◈. Since the anaphora after regeneration would be resolved to the mixture of oil and onion, which is marked with ●, we consider that no

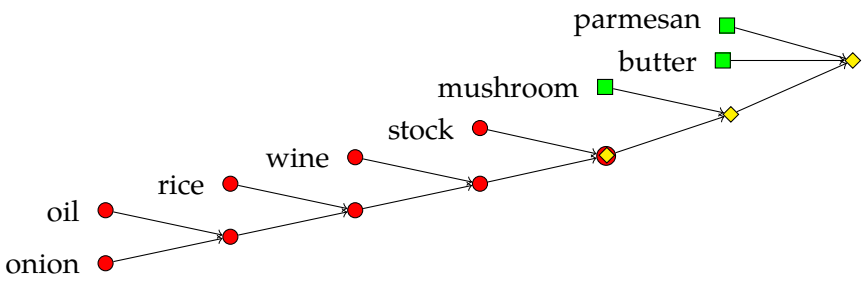


Figure 4.3 – A directed acyclic graph tracing object transformation in the risotto recipe.

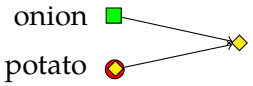


Figure 4.4 – A directed acyclic graph tracing object transformation in the sample example recipe.

explicitation is needed.

This approach is also successful when applied to the earlier example, "Cut the onions, then peel the potatoes and boil them. Combine", for which the graph is shown in figure 4.4. After movement, the anaphora would be resolved to onion, which is marked with ■, and is therefore not acceptable, leading to explicitation.

4.3 Related Work

The problem of text reuse in textual case-based reasoning has been addressed in different manners. For instance, Gervás et al. (2007) suggest using a natural language generation system following the adaptation of the underlying formal representation of the textual solution. Adeyanju et al. (2009) reuses textual solutions by aggregating small chunks of text from different solutions, a type of compositional adaptation. While Lamontagne and Lapalme's (2004)'s proposal does not include an adaptation stage, it does help the end user in adapting the textual solution themself by identifying phrases that should be modified or could be removed altogether.

As far as we know, though, ours is one of the first work addressing the issue or regenerating text in order to change an aspect of its meaning.

On the other hand, there has been much work on text regeneration, also known as text–to–text generation, in which the goal was to produce new text based on and having the same meaning as existing text. This task is, in

essence, paraphrasing, and has many applications, including text simplification or summarisation, and question answering.

A text simplification system attempts to create paraphrases that obey certain stylistic constraints, typically to maximise the text's score in a given readability metric. Different approaches exist, for instance based on regeneration as defined herein, or on analysis followed by generation. Siddharthan (2011) has demonstrated that, while both solutions have merits, the one based on regeneration is more robust and gives better result in terms of syntax.

A summarisation system based on regeneration usually extracts salient sentences from a text—or from multiple texts—based on a variety of heuristics or models, and assemble them while making small adjustments to ensure the resulting text reads fluently. Many systems use a comparable approach, albeit Radev and McKeown (1998) were probably the first ones to define a formalisation of theirs.

Finally, question answering, a task in which questions asked by users are matched with available texts, requires the use of paraphrasing to handle situations in which the question and the answer are formulated differently—e.g. the question is "Who *is the author of* the 'Star Spangled Banner'?" and one text document contains the sentence "Francis Scott Key *wrote* the 'Star Spangled Banner' in 1814" (Lin and Pantel, 2001, emphasis added). Most symbolic approach to this questions, such as Lin and Pantel's, work with paraphrasing rules learnt from monolingual aligned corpora.

Additionally, Valls and Ontañón (2012) propose a case-based reasoning approach to surface realisation in a natural language generation system that could arguably be categorised as text regeneration. In their approach, a sentence is retrieved and a phrase substitution is performed. Because the modifications occur at the sentence level, though, only a small subset of the difficulties inherent to regeneration is addressed in their work.

Chapter 5

Extension to Spatial Problems

In the introduction, we mentioned that one of the advantages of using qualitative algebras is the possibility of applying the solutions described herein to different types of spatial or temporal problems, beyond procedures.

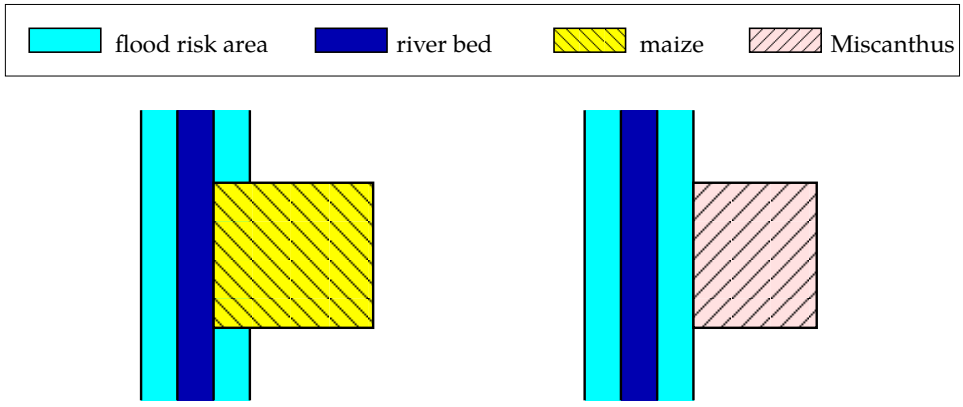
In this chapter, we present a small example extending our approach to perform spatial reasoning in order to solve a farming problem: crop allocation. First, the problem is introduced, along with an example and the expected solution. We present spatial algebras appropriate for this sort of problem and propose a formalisation of the example as a qualitative constraint network (QCN) using the RCC8 algebra. Then, we apply the solution proposed in chapter 3 to it and show that the result we obtain is the one we expected. Finally, we discuss the possibilities and the limitations related to this application.

This work was introduced in Dufour-Lussier et al. (2012b).

5.1 Crop Allocation Problem

Spatial adaptation is illustrated using the example of Miscanthus allocation practices in agriculture, following the research work of Martin et al. (2012). Miscanthus is a perennial grass currently promoted as a renewable source of energy in Europe to produce high yield of biomass with low input (Clifton-Brown et al., 2004). Its potential to reduce greenhouse gas emission is dependent to its spatial allocation into farmlands, as land use change can enhance greenhouse gas emissions (Hillier et al., 2009) and food/non-food competition (Karp and Richter, 2011). Therefore modelling the changes in land usage into farmlands is of great interest.

CBR can be used to model Miscanthus spatial allocation. The problem is the crop production requirements and the farm description, and the solution is a crop spatial allocation. A farm description is defined by a cropping plan (the crop proportions/allocations into farmland) and by the spatial



(a) Source case: a maize farm. (b) Adapted case: a Miscanthus farm.

Figure 5.1 – Crop spatial allocation example.

farmland features (e.g. the spatial relations of plots with buildings, woodland and rivers).

In this example, illustrated by figure 5.1, a farmer who wants to cultivate Miscanthus is considered. A case corresponding to a maize farm could be retrieved (figure 5.1a), and expert knowledge which identify similarities in Miscanthus and maize allocation requirements regarding temperature and soil moisture (Zub and Brancourt-Hulmel, 2010).

Replacing maize with Miscanthus (which is usually harvested from February to March in France) comes with a spatial constraint in the agronomic domain knowledge. Because access to plots by harvesting machinery is impaired by excess soil water in winter, Miscanthus must not be allocated near a river, in a flood-risk area, whereas maize can be planted up to a legal 5 metres from rivers.

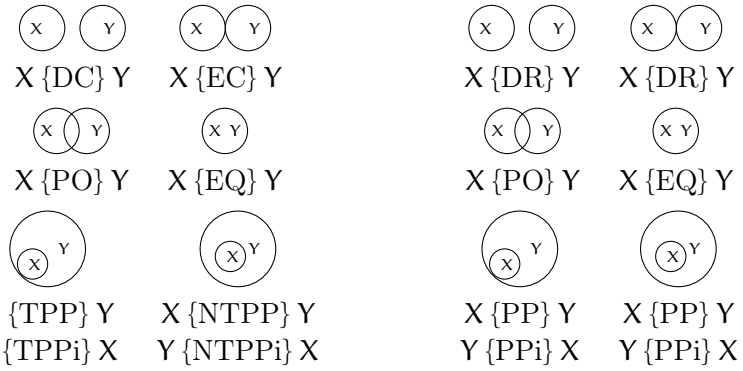
Therefore, it is expected that the adaptation process would not only replace maize by Miscanthus, but also reduce the size of the plot so that it does not overlap with the flood plain of any nearby river (figure 5.1b).

5.2 Qualitative Spatial Algebras

In chapter 1, we introduced qualitative algebras that can be used to represent the relations between variables to be interpreted over the rational numbers $\mathbb{Q}$. Some of those can easily be extended to represent the relations between variables over $\mathbb{Q}^n$.

Balbiani et al. (1998) extended interval algebra to the second dimension, yielding a calculus for relations between rectangles in $\mathbb{Q}^2$, in which the 169 base relations are pairs of Allen relations:

$$\mathfrak{B}_{\text{rectangles}} = \mathfrak{B}^2_{\text{Allen}} = \{(r, s) \mid r, s \in \mathfrak{B}_{\text{Allen}}\} \tag{5.1}$$



(a) RCC8. (b) RCC5.

Figure 5.2 – The relations of calculi RCC5 and RCC8.

This can be extended to the relations between n-blocks in $\mathbb{Q}^n$:

$$\mathfrak{B}_{n\text{-blocks}} = \mathfrak{B}^n_{\text{Allen}} \tag{5.2}$$

Randell and Cohn (1989) took a different approach, proposing a family of algebras named region connection calculi (RCC), in which the primitives are regions in space rather than dimensionless points. All the relations are derived from Clarke's (1981) connection relation, which is reflexive and symmetric. From this relation, Bennett (1994) derive the five relations of RCC5 shown in figure 5.2b, Randell, Cui, and Cohn (1992) derive the eight relations of RCC8 shown in figure 5.2a. Other, more complex RCC calculi exist, such as RCC15, in which the convex hull of the regions is taken into account to further specialise EC and DC.

In the farming example introduced above, the main features that need to be included in a case representation are the land plots and features (rivers, forests, soil types, etc.). Those can be represented as convex regions of the planes, but not necessarily as evenly oriented rectangles. Therefore, of the formalisms introduced, only the RCC family of calculi could be appropriate. Moreover, we think that the connection without overlap of regions is relevant, since it is important in farming to distinguish between features that touch and features that are close together but with a certain distance—the mandatory 5 metre uncultivated strip along water could be an example of this. This rules out RCC5. On the other hand, the example does not indicate any necessity of distinguishing neatly between different RCC8 relations, therefore we propose using RCC8.

To address the fact that there is a difference in possible agricultural uses between the bed of the river and the zone with flood risks, it is broken in two regions, 'low water channel' and 'flood plain', such that the former is a proper part of the latter and that their boundaries do not touch. This is

expressed in RCC8 as

$$\text{'low water channel'}\ \{\mathrm{NTPP}\}\ \text{'flood plain'} \quad .$$

To simplify the formalisation of the problem, we can postulate that the not cultivable 5-metre strip along the river is part of the 'low water channel' region. Because the maize plot will have to be replaced by a Miscanthus plot, it makes sense to parametrise this variable as 'plot(maize)'. The fact that a maize plot is adjacent to a river is represented as

$$\text{'plot(maize)'}\ \{\mathrm{EC}\}\ \text{'low water channel'} \quad .$$

A farmer wishes to cultivate Miscanthus in a similar setting, prompting the retrieval of the farm case just described. A substitution must be applied:

$$\sigma = \text{maize} \rightsquigarrow \text{Miscanthus} = \text{maize} \rightsquigarrow x\ ;\ x \rightsquigarrow \text{Miscanthus} \quad .$$

An important knowledge about Miscanthus is that it must not be cultivated in a zone susceptible to flooding, which can be expressed as

$$\text{'plot(Miscanthus)'}\ \{\mathrm{DC},\ \mathrm{EC}\}\ \text{'flood plain'} \quad .$$

5.3 Case Adaptation in RCC8

Keeping in mind that equation 3.15 defined the formula for revision-based adaptation with qualitative algebras and substitution as

$$\mathtt{DK} \wedge \alpha(\mathtt{Source}) \dot{+} \mathtt{DK} \wedge \mathtt{Target} \wedge \mathcal{N}_\rho$$

where DK is the domain knowledge, $\alpha(\mathtt{Source})$ is the source in which the parameters to be substituted have been abstracted (maize $\rightsquigarrow$ x), and $\mathcal{N}_\rho$ contains the constraints necessary for the refinement (x $\rightsquigarrow$ Miscanthus), in this example, ψ contains the constraints

$$\begin{aligned} C_{\mathtt{DK}} &= \{\text{'plot(Miscanthus)'}\ \{\mathrm{DC},\ \mathrm{EC}\}\ \text{'flood\_plain'}\} \\ C_{\alpha(\mathtt{Source})} &= \left\{ \begin{array}{l} \text{'low\_water\_channel'}\ \{\mathrm{NTPP}\}\ \text{'flood\_plain'} \\ \text{'plot(x)'}\ \{\mathrm{EC}\}\ \text{'low\_water\_channel'} \end{array} \right\} \end{aligned}$$

and μ contains the constraints

$$\begin{aligned} C_{\mathtt{DK}} &= \{\text{'plot(Miscanthus)'}\ \{\mathrm{DC},\ \mathrm{EC}\}\ \text{'flood\_plain'}\} \\ C_{\mathtt{Target}} &= \{\text{'low\_water\_channel'}\ \{\mathrm{NTPP}\}\ \text{'flood\_plain'}\} \\ C_{\rho} &= \{\text{'plot(x)'}\ \{\mathrm{EQ}\}\ \text{'plot(Miscanthus)'}\} \end{aligned}$$

The first step in the algorithm is to add missing variables and constraints. In the example, all four variables are present in both QCNs, but

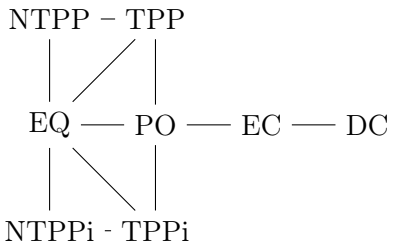


Figure 5.3 – The relation neighbourhood graph of RCC8 is the graphical representation of the direct topological transitions between relations given by Randell et al. (1992).

some relations are missing, e.g. between 'plot(Miscanthus)' and 'low_water_channel'. A constraint

'plot(Miscanthus)' {DC, EC, PO, TPP, NTPP, TPPi, NTPPi, EQ} 'low water channel'

is therefore added to both ψ and μ.

This manipulation may complexify the QCNs, which is part of the reason why computing the algebraic closure is interesting. Here, the amount of potential scenarios is reduced from 1024 to 16 for ψ, and from 1024 to 4 for μ: in both cases, 'low_water_channel' {DC} 'plot(Miscanthus)' can be deduced, which, given the other constraints, entails that 'plot(x)' {EQ} 'plot(Miscanthus)' cannot hold in ψ, and entails 'plot(x)' {DC,EC} 'flood_plain' in μ.

Then, a search is performed on the scenarios of μ, parametrised with their distance towards ψ, which is computed thanks to the relation neighbourhood graph given in figure 5.3. Only one scenario $\mathcal{T}$ of μ is found at the minimal distance, which is 3: $\psi \dot{+} \mu = \{\mathcal{T}\} = \{(V_{\mathcal{T}}, C_{\mathcal{T}})\}$ with

$$C_{\mathcal{T}} = \left\{ \begin{array}{l} \text{'low\_water\_channel' \{NTPP\} 'flood\_plain'} \\ \text{'low\_water\_channel' \{DC\} 'plot(x)'} \\ \text{'low\_water\_channel' \{DC\} 'plot(Miscanthus)'} \\ \text{'flood\_plain' \{EC\} 'plot(x)'} \\ \text{'flood\_plain' \{EC\} 'plot(Miscanthus)'} \\ \text{'plot(x)' \{EQ\} 'plot(Miscanthus)'} \end{array} \right\}$$

The distance is the sum of the following replacements: EC becomes DC between 'plot(x)' and 'low water channel' (d = 1), {TPP, NTPP, PO} becomes EC between 'flood plain' and 'plot(x)' (d = 1), and {DC, EC, NTPPi, PO} becomes EQ between 'plot(x)' and 'plot(Miscanthus)' (d = 1).

In this scenario, the region 'plot(x)' was reduced in order not to overlap with 'flood plain' as it was equated to 'plot(Miscanthus)'. This corresponds to the allocation shown in figure 5.1b. It can be seen that the modification is indeed minimal, as the plot becomes externally connected to the flood

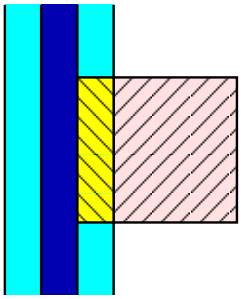


Figure 5.4 – Another adapted case: a maize *and* Miscanthus farm.

plain, maximising the area used for Miscanthus cultivation. For instance, a result including

$$\text{'flood plain'} \{\text{DC}\} \text{'plot(Miscanthus)'}$$

would have been consistent with the domain knowledge but would not have constituted a minimal modification of ψ. Therefore, the adaptation is successful.

5.4 Discussion

This example has been simplified, but the proposed solution could work with more complex cases, for instance including more plots and regions representing features such as soil types. The retrieval has been taken for granted but, with more complete case descriptions, could be implemented following the suggestions put forward in section 3.4.

An arguably more satisfactory solution to the problem discussed above would be breaking up the maize plot in two parts in order to retain maize production in the flood plain and thereby ensure maximum farmland productivity. This solution is shown in figure 5.4. Two solutions to implement this idea can be suggested.

First, the algorithm that searches the models of the QCN μ could be modified in such a way as to be able to create or delete variables from the QCN, under certain conditions and triggering a certain cost. This solution would also benefit our main application.

Second, additional steps could be implemented, before the revision, to break plots up according to certain criteria. For instance, the plots could be broken up following the other region borders. In the case shown in figure 5.1a, the maize plot would be broken up in two plots such that

$$\text{'plot(maize}_1\text{)'} \{\text{TPP}\} \text{'flood plain'}$$
$$\text{'plot(maize}_2\text{)'} \{\text{EC}\} \text{'flood plain'}$$
$$\text{'plot(maize}_1\text{)'} \{\text{EC}\} \text{'plot(maize}_2\text{)'}$$

Applying either $\sigma = \text{maize}_1 \rightsquigarrow$ Miscanthus or $\sigma = \text{maize}_2 \rightsquigarrow$ Miscanthus, the adaptation returns five scenarios. One corresponds to the allocation shown in figure 5.4. The other ones are similar.

Even this simplified example shows that different substitutions become possible if regions are broken up. One solution would be to attempt the revision with each possible substitution and retain only the ones that minimise the distance to ψ.

Chapter 6

Implementation

This chapter presents CRAQPOT, the software implementation of the model we propose for textual procedure reuse. CRAQPOT is a Case-based Reasoning Adaptor of Qualitative Procedures Over Texts which provides an interface to obtain recipes in response to any query.

The processes for case acquisition, for formal case adaptation and for textual adaptation, as well as the user interface, are all implemented separately, and will therefore be presented individually in sections 6.1, 6.2, 6.3 and 6.4. While the process of obtaining domain knowledge is not within the scope of this thesis, having some is essential in order to test the ideas put forward herein. Section 6.5 explains how the simulated knowledge we used in our implementation was acquired.

The level of genericity in the implementations is variable: while the library providing revision-based case adaptation is entirely generic and can be used for any application using any qualitative algebra, the libraries for case acquisition and for textual adaptation could be ported to different domains, but with a greater effort.

The retrieval process suggested in section 3.4 is not implemented. Rather, CRAQPOT relies on a simplified but faster reimplementation of TUUURBINE (Gaillard et al., 2014), a generic, ontology-guided case-based inference engine, using WIKITAAABLE[13] (Badra et al., 2009; Cordier et al., 2013) as its knowledge base.

The implementation of the case acquisition is discussed in Dufour-Lussier et al. (2014c), that of revision-based adaptation for qualitative algebras is introduced in Dufour-Lussier et al. (2012b) and updated in Dufour-Lussier et al. (2013). The implementation of the textual adaptation and of the user interface were not the object of prior publications, even though a previous textual reuse algorithm was introduced in Dufour-Lussier et al. (2010)

13. `http://wikitaaable.loria.fr`

6.1 Case Acquisition

In chapter 2, we discussed existing systems which can be used to develop natural language understanding systems. Nonetheless, to make it easier to experiment with different methods of applying each step in the tool chain, it seemed logical at implementation type to create a text analysis toolchain from scratch. At this point though, reimplementing the system using one of the systems presented in chapter 2 would make it easier to port it to different application domains.

This program takes as input text representing the title, the ingredients, and the preparation steps of a recipe, usually provided in an XML format defined by the organisers of the Computer Cooking Contest (CCC).

It gives different output, depending on the needs of the calling program. Most importantly with respect to what is discussed in this thesis, it outputs back the text with an enriched XML annotation mixing the CCC tags with additional information about actions and their timing, encoded using the TimeML (Pustejovsky et al., 2003) standard and custom tags and attributes.

6.1.1 Standard NLP Toolchain

Most of the standard steps were implemented using *regexes* in Perl. Regexes in Perl and Perl-compatible implementations, while inspired by Kleene's (1956) regular expressions, are not actually regular. Indeed, backreferences—using a variable inside a regex to match a part of the same regex—and zero-width assertions—allowing a match only before or after a given sub-regex, without actually matching the said sub-regex—make it possible to define irregular languages. For instance, using a backreference, the language of squares

$$\{ww \mid w \in \Sigma^*\} \quad ,$$

which can be shown not to be regular or even context-free by the pumping lemma, is recognised by Perl regex $/(.*)\backslash 1/$.

While the languages defined by regexes have been proven to be context-sensitive (Câmpeanu et al., 2003), the matching is usually implemented using so-called backtracking, "traditional" nondeterministic finite automata (NFA)—which are technically not "real" NFA. The complexity of the matching operation is $O(2^s)$ where s is the amount of states in the NFA but, in practice, traditional NFA are normally efficient and stop as soon as a match is found, making optimisations easy to implement (Friedl, 2006).

Named entity recognition, which includes number recognition, tokenisation and segmentation are implemented using hand-written, irregular regexes. Shallow syntactic analysis is implemented using a regular phrase structure grammar, that is, a grammar using rewrite rules that can be implemented by a (true) NFA. Being implemented as a regex as well, the parser

does not generate ambiguity because it forms the largest phrases allowed by the grammar. An alternative chunker is provided, which is implemented as a "RegexpParser" object from NLTK (Bird, 2006). Other, less central functions are formulated as regexes as well, such as the stemmer—which finds the root of a given word, removing for instance verbal or plural endings.

The part-of-speech tagger uses a default tag for each word form and context-sensitive rewrite rules, as defined by Brill (1992). Both were induced from an annotated corpus then saved directly as associative arrays in Perl. The lexicon of recipes being limited, it was possible to provide a default tag manually for words that appeared in the recipe corpus but not in the annotated part. No such word appear in the test corpus, therefore the quantitative results provided remain valid. A Brill tagger tags an n-word text using r rules in time $O(n \cdot r)$.

After those steps are completed, the text is traversed many times to annotate relevant features. Some of those traversals could technically be grouped, but each pass only takes $O(t)$ where t is the amount of tokens in the text, which therefore does not add much to the processing time, and keeping them separated makes the implementation easier to maintain.

One pass recognises words representing ingredients and annotates them with respect to the ingredient list. For instance, the word "oil" is recognised to mean "olive oil" if this is the only type of oil found in the ingredient list. In an optimal implementation, we reckon that this should make use of the food ontology, but in the actual implementation, it is strictly based on lexicon. For instance, again, the word "fat" would not be recognised to mean "olive oil" by this process even if it was the only type of fat. On the other hand, the anaphora resolution process would put this right at a later step.

Other passes identify candidate lexical anaphoras: lexical variants of concepts in the food ontology for those we called universal references and words corresponding to previously computed sets for those we called existential references—as well as some additional types such as the words "next n ingredients", where n can be interpreted as an integer.

Verbs indicating cooking actions are recognised, and their subcategorisation frame, action class, and mood are indicated. Then, using the output of the chunker, the phrases in which to look for verb complements are identified, and actual candidate complements representing ingredients are found. Temporal modifiers are also identified at this time.

6.1.2 Event Chain Building

The CCC::EventChain module takes the analysis over when the standard analysis is done. It is responsible for incrementally building the case representation by resolving the anaphoras, as described in sections 2.2 and 2.3. Once the model is clearly defined, computation is in fact straight-

forward.

An initial set of foods is computed from the ingredient lists, then the first action is analysed. Each candidate complement is taken and, if it can be mapped in a straightforward way to a member of the initial set of foods, this food is considered an input of the action. If a candidate has been marked as an anaphora candidate, resolution is attempted by matching against all the members in the set of foods. Finally, all input foods are removed from the set, and the output is computed and added to the set. The output also becomes the set of active foods, which will be used as an input for the next action if it is missing one complement according to its subcategorisation frame.

The complexity of the anaphora resolution is linear with the size of the set of available foods. In theory, the existence of splitting actions makes the size of this set unbounded. Empirically though, we see that the set initially has the size of the ingredient list i, and its size steadily decreases to reach 1.

6.2 Revision-Based Adaptation

As demonstrated in chapter 3, adapting cases represented using a qualitative algebra can be as simple as performing a distance-based belief revision on the source and target cases.

While much of the theoretical work making the implementation of such an operator possible has already been performed by Condotta et al. (2008), there is no available implementation at this time. We therefore created two implementations.

Given ψ and μ, the first method removes interpretations of μ until all the models are as close to ψ as possible, while the second method simultaneously relaxes and contracts the interpretation of ψ until it is a subset of the interpretation of μ. The first method honours the postulates defined by Katsuno and Mendelzon (1991) but has exponential complexity. The second method is polynomial, but may not honour all the postulates.

Subsection 6.2.1 describes the data preparation step common to both methods. Then, the two methods are presented in subsection 6.2.2 and subsection 6.2.3. Finally, subsection 6.2.4 gives implementation details for both methods.

6.2.1 Data Preparation

The program takes as input a source case `Source`, a target problem `Target`, knowledge domain `DK`, and a set of substitutions σ. In CRAQPOT, `Source` is a QCN extracted from the XML output by the natural language understanding process, and `Target` is actually empty as in the example

detailed in section 3.3 The origin of the domain knowledge is discussed in section 6.5.

Quoting equation 3.15, the adapted case is:

$$\mathtt{DK} \wedge \alpha(\mathtt{Source}) \dot{+} \mathtt{DK} \wedge \mathtt{Target} \wedge \mathcal{N}_\rho$$

QCN conjunction is defined in section 3.3. The set of variables of the conjunction of two QCNs is the union of the set of variables of each. In practice, this means that any variable present in one QCN has to be created in the other if it does not exist already. Those variables initially gets the uninformative relation ? with respect to every other variable. Then, for each constraint, the intersection of the base relations present in both QCNs is retained. If any such intersection is the empty set, then the conjunction is inconsistent—in practice, the case or the problem is not consistent with the domain knowledge—and the adaptation must fail.

Substitution is defined in section 3.3. The implementation is straightforward. Each atomic substitution $p \rightsquigarrow q$ is divided into an abstraction $p \rightsquigarrow x_p$ and a refinement $x_p \rightsquigarrow q$. For every abstraction $p \rightsquigarrow x_p$, each instance of concrete parameter p in `Source` is replaced by an unused abstract parameter to obtain $\alpha(\mathtt{Source})$. For every refinement $x_p \rightsquigarrow q$ and for every occurrence $\nu(x_p)$ of abstract parameter x_p in `Target`, a constraint $\nu(x_p)$ EQ $\nu(q)$ is created and conjoined to $\mathtt{DK} \wedge \mathtt{Target}$.

In order for the belief revision to be carried, it is necessary, once more, that any variable present in one QCN be created in the other if it does not exist already. This process is carried on between the terms of the revision in the same way as between the terms of a conjunction.

6.2.2 Model Search

As described before, the main function performed by an application computing the result of a revision $\psi \dot{+} \mu$ is a search through the models of μ for those closest to ψ. We chose to implement this search using an A* search algorithm (Hart et al., 1968).

We start with μ, pick one constraint, and strengthen it by retaining only one base relation. Then we pick another constraint to strengthen, and repeat the process until a scenario is found. If the scenario is path-consistent, a solution has been found. Otherwise, it is necessary to backtrack up the search tree and either chose a different base relation to retain, or a different constraint to strengthen.

Each constraint strengthening has a cost associated to it, because it may bring μ further away from ψ. It also has a carries a benefit, because after strengthening a constraint, less constraints remain to be modified in a way that may take μ away from ψ. This information is used to inform the order in which the search tree is explored: no branch is followed if there exists a possibility that a different branch may bring a better result.

The actual cost paid can be computed exactly, whereas it is only possible to estimate the minimum cost remaining to be paid for the constraints that have not be dealt with at any given point in time. The former is computed using a cost function, and the latter is computed using a heuristic function. The search is guided by an aggregation of those two functions. In our implementation, the aggregation is a sum. As the process moves on, the cost gains importance over the heuristic, and the guidance provided is more exact. When a scenario is found, the exact distance between this scenario and ψ is known. Moreover, if the heuristic is valid, i.e. if it never overestimates the distance left to be covered, it is guaranteed that the first scenario that is found is at least as good as any other, and the search can be stopped.

To limit the size of the search tree, we apply algebraic closure at each step of the process: the search actually starts with the algebraic closure of μ and, each time a constraint is strengthened, algebraic closure is applied to the result again. Empirical evidence shows that, for QCNs larger than four variables, the time lost by computing algebraic closure is less than the time gained by not having to consider every possible base relation on all constraints. Moreover, because a path-inconsistent QCN is guaranteed to be inconsistent, unproductive search branches can sometimes be abandoned early when path-consistency is enforced.

Formally, an A* algorithm is defined by its initial, successor and final states, and by its cost (g) and heuristic (h) function.

Any state e of our algorithm is a QCN $(\mathcal{V}, \mathcal{C})$. The initial state e_0 is the algebraic closure of the QCN $\mu = (\mathcal{V}, \mathcal{C})$. A final state is a model of μ, i.e. a scenario (each constraint has only one base relation) which is path-consistent.

A successor state $e'_\circ = (\mathcal{V}'_\circ, \mathcal{C}'_\circ)$ of a state $e = (\mathcal{V}, \mathcal{C})$ is the algebraic closure of a QCN $e' = (\mathcal{V}', \mathcal{C}')$, where $\mathcal{V} = \mathcal{V}' = \mathcal{V}'_\circ$, and $\mathcal{C}$ is identical to $\mathcal{C}'$ except for one constraint in which only one base relation was retained. In practice, at each branching of the search tree, all possible successors to a given state are computed and inserted in a priority queue.

The algebraic closure is computed using Vilain and Kautz's (1986) adaptation of Mackworth's (1977) constraint propagation algorithm, shown in algorithms 1 and 2. Other algorithms have been proposed by van Beek (1990, 1992); Dechter et al. (1991); Bessière (1996). The more recent algorithms tend to be faster in experiments on random QCNs, but they have given unsatisfactory results when applied to our low-density (i.e. few asserted constraints), high-tightness (i.e. few base relations on each asserted constraint) QCNs.

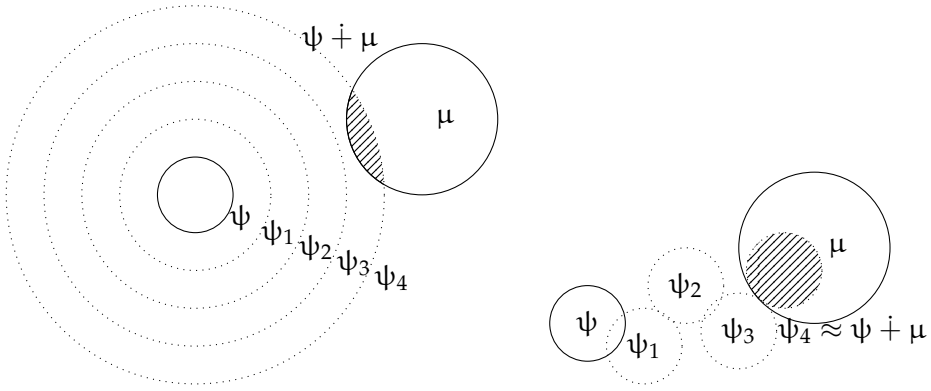
The path cost function uses the QCN ψ, which has the same variables as μ. As defined in chapter 1, the distance between two base relations is their distance in the relation neighbourhood graph, and the distance between two complex relations is the minimum of the distance between any two of their base relations. The path cost function is the sum of the distances between each constraint of μ that was processed and the corresponding constraint of

Algorithm 1 Constraint propagation algorithm (Vilain and Kautz, 1986).

▷ Takes a QCN; enforce algebraic closure as a side effect; returns status
function PROPAGATECONSTRAINTS($\mathcal{N} = (\mathcal{V}, \mathcal{C})$)
    $Q \leftarrow \{(x, y) \mid x \in \mathcal{V}$ and $y \in \mathcal{V}$ and $x < y\}$ ▷ Arbitrary order
    while $Q \neq \emptyset$ **do**
        $(x, y) \leftarrow$ next element of Q
        for $z \in \mathcal{V}$ such that $z \neq x$ and $z \neq y$ **do**
            $r_1 \leftarrow$ FIXCONSTRAINT(C_{xy}, C_{yz}, C_{xz})
            if $r_1 =$ CONTRACTED **then**
                add (x, z) to Q
            else if $r_1 =$ INCONSISTENT **then**
                return INCONSISTENT
            $r_2 \leftarrow$ FIXCONSTRAINT(C_{zx}, C_{xy}, C_{zy})
            if $r_2 =$ CONTRACTED **then**
                add (z, y) to Q
            else if $r_2 =$ INCONSISTENT **then**
                return INCONSISTENT
    return SUCCESS;

Algorithm 2 Constraint propagation algorithm: local path-consistency.

▷ Takes three constraints; enforce 3-path-consistency as a side effect; returns status
function FIXCONSTRAINT(C_{xy}, C_{yz}, C_{xz})
    $C'_{xz} \leftarrow C_{yz} \circ C_{xy}$;
    if $C_{xz} \subseteq C'_{xz}$ **then**
        return NO-CHANGE-NEEDED
    if $C_{xz} \cap C'_{xz} = \emptyset$ **then**
        return INCONSISTENT
    $C_{xz} \leftarrow C_{xz} \cap C'_{xz}$
    return CONTRACTED



(a) Grove visualisation of revision.

(b) Revision approximation.

Figure 6.1 – Grove's viewpoint compared to the approximation algorithm.

ψ.

The heuristic function uses the same definition of the distance, but computes the minimal possible cost to be incurred by the constraints yet to be processed. For each unprocessed constraint in μ, its distance to the corresponding constraint on ψ is computed as above. The sum of those is the heuristic. This heuristic is provably valid, i.e. it cannot overestimate the remaining cost, because the minimum of sums is never less than the sum of minimums.

In the worst case, all nodes corresponding to scenarii of μ will need to be expanded. Given a QCN $(\mathcal{V}, \mathcal{C})$ with constraints expressed using the set of base relations $\mathfrak{B}$, the amount of scenarii is of the order of $O\left(|\mathfrak{B}|^{\frac{|\mathcal{V}|\cdot(|\mathcal{V}|-1)}{2}}\right)$.

6.2.3 Approximating Revision with Repair Propagation

One way to model the belief revision $\psi \dot{+} \mu$, proposed by Grove (1988), is to show a system of concentric spheres around ψ, each corresponding to a certain level of relaxation. The result of the revision, in this system, is the intersection of μ and of the smallest sphere around ψ such that the intersection is not empty. This is illustrated in figure 6.1a.

We propose an alternative revision algorithm, more efficient than our model search, inspired by this viewpoint, which also takes advantage of the existing constraint propagation algorithm. Since propagation requires cubic time in the worst case, and given that it will be applied $\frac{|\mathcal{V}|\cdot(|\mathcal{V}|-1)}{2}$ times in the worst case—once for each constraint asserted in μ—the revision will complete in less than $O(|\mathcal{V}|^5)$, a significant speed improvement. The tradeoff is that neither correctness nor completeness has been proven for the alternative algorithm. Indeed, in algebras for which path-consistency is not a sufficient condition for consistency, the result might be inconsistent.

Intuitively, the principle of this approximation algorithm is that constraints from μ are added to and propagated in ψ, as shown in algorithm 3. In case $\psi \wedge \mu$ is not consistent, the propagation algorithm will meet with impossible constraints. Therefore, it must be modified so that, instead of rejecting the QCN as inconsistent when this happens, it relaxes the faulty constraint.

This implies that ψ is alternatively relaxed, as expected in Grove's model, and strengthened. The stopping point of the algorithm will thus not be when the set of models of the relaxation of ψ has an intersection with the set of models of μ, but rather when the set of models of the deformation of ψ is a subset of the set of models of μ. This is illustrated in figure 6.1b.

To ensure that the success postulate is respected—i.e. the result of the revision entails the new knowledge—a relaxation limit is enforced: $C_{\psi,xy}$ can never be relaxed in a way such that it contains relations not found in $C_{\mu,xy}$. Moreover, to ensure that the subexpansion postulate is respected—i.e. ψ is modified as little as possible—it is necessary *not* to compute the algebraic closure of μ, and to only propagate the beliefs explicitely asserted in μ onto ψ. This, on the other hand, sacrifices the extensionality postulate responsible for guaranteeing the irrelevance of syntax. The outline of the modified propagation algorithm is shown as algorithm 4, continued as algorithm 5.

The function RELAX is not shown. Two ways of computing it are outlined:

— Replace all the base relations used in the constraint with the base relations that are at relation-to-relation distance $d = 1$. This will enforce the subexpansion postulate.

— Immediately replace the constraint C_{xz} with the deduced constraint $C_{yz} \circ C_{xy}$. This does not respect the subexpansion postulate with respect to the definition of the distance given earlier, but has the effect of changing less constraints.

When used to revise variables representing actions that are mostly sequential, the first method tends to make small changes to many variables, in such a way that the actions tend to overlap. The second method makes more radical changes, but preserves the sequentiality better. For this reason, the second method is used in CRAQPOT.

6.2.4 Technical Details

The functionalities described in this section were implemented as a Perl module QA::QCN. It uses modules descended from QA::Algebra to define the algebras. Allen, $\mathcal{INDU}$ and RCC8 are provided. Two interfaces are provided: a command-line interface in Perl which parses QCNs expressed in text files, and a Java library that uses the Perl module as a backend. The modules are being rewritten in Java and will soon offer better performance. It will be possible to define additional algebras as XML files rather than

Algorithm 3 Fast, non-AGM revision algorithm.

▷ Takes two QCNs; returns a revised QCN
function ReviseApproximately($\psi = (\mathcal{V}, \mathcal{C}_\psi)$, $\mu = (\mathcal{V}, \mathcal{C}_\mu)$)
    $\varepsilon \leftarrow (\mathcal{V}, \mathcal{C}_\psi)$ ▷ Let us name the result ε
    $\mathcal{C}^{\mathrm{L}}_\varepsilon \leftarrow \emptyset$ ▷ Initialise the set of *locked* constraints
    for $(x, y) \in \mathcal{V}$ such that $x < y$ **do**
        $C_0 \leftarrow C_{\varepsilon,xy}$
        $C_1 \leftarrow C_{\mu,xy} \cap C_0$
        while $C_1 = \emptyset$ **do**
            $C_0 \leftarrow \textsc{Relax}(C_0)$
            $C_1 \leftarrow C_{\mu,xy} \cap C_0$
        $C_{\varepsilon,xy} \leftarrow C_1$
        $C^{\mathrm{L}}_{\varepsilon,xy} \leftarrow C_{\mu,xy}$
        PropagateRepair(ε, $\mathcal{C}^{\mathrm{L}}_\varepsilon$)
    return ε

Algorithm 4 Modified propagation algorithm: global path-consistency.

▷ Takes a QCN and a set of mandatory constraints; enforces algebraic closure as a side effect; returns status
function PropagateRepair($\mathcal{N} = (\mathcal{V}, \mathcal{C})$, $\mathcal{C}^{\mathrm{L}}$)
    $Q \leftarrow \{(x, y) \mid x \in \mathcal{V} \text{ and } y \in \mathcal{V} \text{ and } x < y\}$
    while $Q \neq \emptyset$ **do**
        $(x, y) \leftarrow$ next element from Q
        for $z \in \mathcal{V}$ such that $z \neq x$ and $z \neq y$ **do**
            $r_1 \leftarrow \textsc{FixConstraintRelaxedly}(C_{xy}, C_{yz}, C_{xz}, C^{\mathrm{L}}_{xz})$
            if $r_1 =$ contracted **then**
                add $\{(x, z)\}$ to Q
            else if $r_1 =$ relaxed **then**
                add $\{(x, y)\}$ to Q
            else if $r_2 =$ inconsistent **then**
                return inconsistent
            $r_2 \leftarrow \textsc{FixConstraintRelaxedly}(C_{zx}, C_{xy}, C_{zy}, C^{\mathrm{L}}_{xz})$
            if $r_2 =$ contracted **then**
                add $\{(z, y)\}$ to Q
            else if $r_2 =$ relaxed **then**
                add $\{(x, y)\}$ to Q
            else if $r_2 =$ inconsistent **then**
                return inconsistent
    return success

Algorithm 5 Modified propagation algorithm: local path-consistency.

```
36:          ▷ Takes three actual and one mandatory constraints; enforce
    3-path-consistency as a side effect; returns status
37: function FixConstraintRelaxedly(C_xy, C_yz, C_xz, C^L_xz)
38:     C'_xz ← C_yz ∘ C_xy;
39:     if C_xz ⊆ C'_xz then
40:         return no-change-needed
41:     if C_xz ∩ C'_xz ≠ ∅ then
42:         C_xz ← C_xz ∩ C'_xz
43:         return contracted
44:     if Relax(C_xz) ∩ C^L_xz ≠ ∅ then
45:         C_xz ← Relax(C_xz) ∩ C^L_xz
46:         return relaxed
47:     return inconsistent
```

having to create new classes.

Our case adaptation and belief revision library for qualitative algebras is available as part of Revisor [14] (Cojan et al., 2013), a collection of open-source belief revision and revision-based adaptation tools for different formalisms.

Hué and Westphal (2012) developed a revision software for qualitative algebras based on Gantner et al.'s (2008) solver, concomitantly with our own work. Their application provides much of the same functionality as ours, but is not publicly available at this time.

6.2.5 Other Possible Optimisations

There exist different kinds of optimisations that could be exploited to make AGM-compliant revision faster. An obvious one would be to exploit tractable subclasses of relations, in different algebras. Different sets of so-called convex relations were found, first by van Beek (1990, 1992), then by Nebel and Bürckert (1995) and others, such that path-consistency is a sufficient condition for consistency in QCNs that use only convex relations.

Knowledge of those subsets can help in consistency checking of arbitrary QCNs. Gantner et al. (2008) have implemented a reasoner that takes advantage of that by trying to divide relations into convex relations in the search tree. This generates a deeper tree, but permits more pruning. Hué and Westphal (2012) have shown that this approach can be reused with good results in model searching for belief revision.

Another avenue of research for optimisations may be "semi-intervals", proposed by Freksa (1992). In procedural texts, some intervals may have only one bound that is actually interesting. For instance, in our implementation, we use neither the upper bound of intervals corresponding to states,

14. `http://revisor.loria.fr`

nor the lower bound of intervals corresponding to actions described in the ingredient list. If it was possible to combine intervals and semi-intervals, our processing could therefore take advantage of the better computational properties of the latter.

6.3 Textual Adaptation

The implementation of the textual adaptation module is still partial and should be considered as a proof of concept. Nonetheless, it has been integrated in the Craqpot web application, and is manifestly contributing to its good evaluation results.

There is no shared annotation resources between the case acquisition and the text adaptation modules at this time. Rather, the annotation rules hard-coded in the case acquisition module were inventoried, and corresponding text adaptation rules were hard-coded in the adaptation module.

The text adaptation algorithm is called by the non-AGM compliant revision algorithm everytime a constraint change is requested, once before starting the propagation, once everytime a constraint is changed, and finally once when the network has stabilised. The first calls are used for bookkeeping only, as the adaptation proper only starts once propagation is over.

All constraint changes are checked one by one, starting with the initial change request, to verify if the change of relation warrants a change in text. Required modification operations are then performed. In changes after the first one, a verification is made as to whether the modification already made upon the text entails the new constraint. For instance, given three events e_1, e_2 and e_3, such that $e_1 < e_2 < e_3$ and a request to move e_3 before e_1, the clause movement will cause the order to become $e_3 < e_1 < e_2$. This entails that $e_3 < e_2$ so, when the propagation will request for e_3 be moved before e_2, no operation will need to be performed.

The operations that are implemented at this time are the ones that appeared to be the most commonly needed:

— Moving clauses, including taking care of punctuation signs and conjunctions that must be added or removed, and of the required capitalisation changes.
— Adding, removing or changing `SIGNAL`s.
— Switching verbs between the imperative and participle moods.

6.4 Interface for End Users

The Craqpot end user interface was implemented as a Facebook application and as a regular web application [15]. A Facebook application is noth-

15. `https://www.loria.fr/barracuda/`

ing more than a website loaded inside an IFRAME element in the user's WWW browser. There are benefits associated with offering a Facebook application rather than an independent website.

First, the effort a user would normally have to provide to create his account and provide personal information is significantly reduced. Upon first launching the application, Facebook will ask the user for an authorisation to share their private information, which is as simple as clicking a button. While the user is then encouraged to fill in their dietary preferences and indicate how good they are at cooking and how well they know English, they can opt to use the application immediately. Thanks to this, moreover, the tasks that are most likely to pose a security risk are handled by Facebook: storing passwords, handling authentication, and communicating with users are examples.

Secondly, Facebook makes the promotion of the application easier. For instance, when the user enters a query and obtains a recipe, their contacts may be informed of this fact and encouraged to use the application themselves. If they decide to use it, it would additionally be possible to ask them to evaluate the recipe obtained by the first user, thereby increasing our confidence in the evaluation.

6.4.1 Flowchart from the User Viewpoint

The application is presented to the user as a "neverending cookbook", i.e. a cookbook that contains an infinite amount of recipe and can therefore be used to answer any non-contradictory query. Unlike on the Taaable website [16], the user does not perceive that a recipe which almost but not entirely answered their query was first retrieved, then adapted.

Like they would in the search engine of any cooking website, the user first specifies the ingredient they want and don't want to use in their recipe. They can additionally specify which type of dish they want, e.g. a dessert dish. They are then presented with exactly one recipe, presented in a classic way, and a three-question evaluation form. The users are asked to evaluate the culinary and the linguistic quality of the recipe. In a future version, it is planned to offer them to see more recipes in answer to their query or to share the result with their contacts, which would be benefitial to us since it would help us gather more evaluations using the same query but different methods.

6.4.2 Flowchart from the Application Viewpoint

The goal of the Facebook application is to evaluate our system by comparing it to other solutions. In total, four different systems are implemented

16. http://www.taaable.fr/

to answer user queries, and one of them is chosen at random whenever a query is received.

The four systems are:

1. A retrieval-only system, which returns a result only if an exact match to the user query is found in the case base. This system is used to provide an upper bound on the culinary and linguistic quality of recipes: even if they are not considered perfect by the user, the recipes returned by any of the other systems cannot possibly be better than those returned by the retrieval-only system, since they all use the same case base.
2. A reimplementation of the solution proposed by Newo et al. (2010). This system implements a minimal, word-based substitution. For instance, if the user asked for a carrot risotto and a mushroom risotto is retrieved, before presenting the result to the user, all occurrences of the word "mushroom" will be replaced by the word "carrot". This system provides a baseline: if we don't obtain a better result, arguably, it is not really worth it implementing the model we propose.
3. The system proposed in Dufour-Lussier et al. (2010). This system does not offer much in the way of temporal adaptation, but it does apply substitution of actions in addition to substitution of ingredients, which is something we consider as an interesting future work for Craqpot. Being able to compare the performances of a system that adapt actions without considering their temporal aspect and one that adapts time without touching at actions will help us confirm whether this is an interesting direction.
4. Obviously, the last system is the one described herein.

Notwithstanding which system is chosen, the next step will be to call Tuuurbine to obtain a source case and a substitution. Then, the adaptation is performed if needed, and the user is presented with the resulting recipe and asked for an evaluation. If the requested system was retrieval-only and no match for the query is found in the case base, an adaptation is performed in order to satisfy the user nonetheless, but their evaluation cannot be used. The same thing happens if an exact match not requiring any adaptation is found in the case base whereas an adaptation method has been chosen: the retrieved recipe is shown to the user, but the evaluation is not used.

6.4.3 Technical Details

The application is, in practice, divided in two different layers.

The first layer is responsible for the website logic. It is a Catalyst FastCGI application responsible for handling the user browser's HTTP requests, generating the HTML code and sending it back. It includes an HTTP client

which connects to the second layer and dispatches the user queries. It receives answers in the CCC XML format and processes them into HTML to send back to the user. It is also responsible for storing the user data, queries and evaluations in a PostgreSQL database.

This second layer handles the CBR logic. It acts as an HTTP server that receives queries from the first layer. It interacts with Tuuurbine, Wiki-Taaable, and all the libraries described in this chapter in order to compute the query results, and sends this back to the first layer.

Therefore a typical query will follow this flow: the user sends their query to the first layer, which relays it to the second layer. The second layer performs the retrieval and the adaptation and sends a recipe back to the first layer. This is formatted as HTML and sent back to the user with an evaluation form. The user fills in the evaluation and sends it back asynchronously by AJAX to the first layer, which stores the information in the database.

6.5 Obtaining Domain Knowledge

The acquisition of domain knowledge for a case-based reasoning application could in itself constitute the topic of a PhD thesis. While the case acquisition method we propose would certainly be useful in acquiring procedure-related knowledge, we consider that this falls outside the scope of this thesis.

As discussed in the introduction, a major advantage of our CBR approach when compared with any classical planning approach to the same problem, is that it will produce useful results without needing complete domain knowledge. As a matter of fact, the approach *will* produce results even with no knowledge at all. On the other hand, those results will not be interesting: making a substitution will not actually cause any clash, and therefore no further adaptation will be suggested.

For instance, given a query for a carrot risotto recipe, the retrieval of a mushroom risotto recipe, and the suggestion of a mushroom $\rightsquigarrow$ carrot substitution, if no information about the cooking time of carrots is known, the system will consider that cooking carrots for two minutes is appropriate.

Therefore, in order to make it possible to evaluate our system, it is necessary to have *some* domain knowledge. The solution we retained is to "fake" domain knowledge by using the prototype retrieval method we proposed in Dufour-Lussier et al. (2010). With this method, given a recipe `Source` and a substitution $\mathtt{p} \rightsquigarrow \mathtt{q}$, a recipe containing `q` is retrieved, such that ingredient `q` in this recipe is treated as much as possible in a similar way as ingredient `p` in `Source`.

The retrieval process is simplified with respect to the one described in the article in that a distance function is used instead of formal concept analysis. The distance between the recipe `Source` containing ingredient `p` and a

candidate recipe containing ingredient q is the amount of actions applied to p in `Source` that are not applied to q in the candidate recipe plus the amount of actions applied to q in the candidate recipe that are not applied to p in `Source`.

As of consequence of this "cheat", the domain knowledge used in our system is stronger than it would be if it was learnt over a set of recipes. For instance, if the carrot recipe most similar to the mushroom risotto is in fact a carrot soup, it may be that our domain knowledge will demand for carrots to be cooked for one hour, and the adaptation result will suffer from this.

From the point of view of calculation time, the effect is difficult to assess. More restrictive temporal knowledge means QCNs with fewer models. For instance,

$$\{\ \text{'cook(carrot)'}\ ?^{=}\ \text{'30 min'}\ \}$$

has 13 models, whereas

$$\left\{\begin{array}{l}\text{'cook(carrot)'}\ ?^{\{>,=\}}\ \text{'10 min'}\\ \text{'cook(carrot)'}\ ?^{\{<,=\}}\ \text{'60 min'}\\ \text{'10 min'}\ ?^{<}\ \text{'60 min'}\end{array}\right\}$$

has 6591. On the other hand, more restrictive knowledge is likely to increase the required adaptation effort and, if the distance between μ and ψ is increased, a greater fraction of the models of μ must be explored during the A^* search before the optimal solution is found.

Chapter 7

Results

Thanks to the implementation of the Craqpot web application, described in the preceding chapter, it is possible to conduct a large-scale evaluation by users. This chapter first presents the global evaluation methodology and preliminary results—based on the first 50 evaluations—in section 7.1. Section 7.3 then presents some ideas that may make it possible to obtain more evaluation results without augmenting the demand on our evaluators. Finally, section 7.2 offers a quantitative evaluation of specific tasks in text annotation.

7.1 Black Box User Evaluation

As mentionned in the preceding chapter, we have implemented an application that can be used to evaluate our work while obtaining recipes. A new user first needs to create an account, which is automatic if they are accessing the application from Facebook. Until they have answered both questions, they will be asked to rate their expertise in cooking and in the English language everytime they access the homepage. They can also immediately make a query, as shown in figure 7.1.

The system has access to two control methods to answer requests:

— A retrieval-only method that performs no adaptation and fails when

Figure 7.1 – Craqpot request interface.

it is not possible to find a recipe that corresponds exactly to the query. Since Craqpot uses the same textual case base, the retrieval system defines the maximum possible scores that could be obtained in text and recipe quality: our proposal is not expected to adapt recipes in such a way that the result is better than the original.

— A method based on a reimplementation of CookIIS text adaptation as described by Newo et al. (2010), which consists in a smart string replacement. This provides a baseline: given the simplicity and efficiency of Newo et al.'s text adaptation, our proposal would be difficult to justify if it did not offer better quality.

A method is selected randomly between the two control methods and the experimental method, and the query is process with this method. If processing fails—which is theoretically possible only with the retrieval method—processing is transferred to another method. The result is then presented to the user, as shown in figure 7.2. The user has no way of knowing which method was used to process their query.

Before the user can make a new query, they are asked to rate their degree of agreement on a 4-point Likert scale—where 1 indicates strong disagreement and 4 indicates strong agreement—with the following statements:

— "This recipe seems tasty." We postulate that this provides the most relevant indicator to evaluate the quality of the *formal* adaptation of the recipe.

— "This text is well written." We postulate that this provides the most relevant indicator to evaluate the quality of the *textual* adaptation of the recipe.

— "This recipe fits my query." We postulate that this provides a very general indicator as to whether the adaptation approach used was appropriate.

The hypotheses we made are:

$\mathbf{H}_1$ Because of the inherent risk of automatic adaptation, Craqpot and CookIIS will offer lower text and recipe quality, and lower fitness than simple retrieval.

$\mathbf{H}_2$ Because it integrates domain knowledge, Craqpot will offer higher recipe quality than CookIIS.

$\mathbf{H}_3$ Because the risk of adaptation is mitigated by finer linguistic processing, Craqpot and CookIIS will offer a similar text quality.

$\mathbf{H}_4$ Because the adaptation is less superficial, Craqpot will leave its users with a better impression that the answer fits the query than CookIIS.

The null hypothesis H_0 is that all three systems are comparable and any difference in score would be the result of chance.

The average normalised ratings for the three indicators and the three methods are detailed in table 7.1. Figure 7.3 shows the probability density

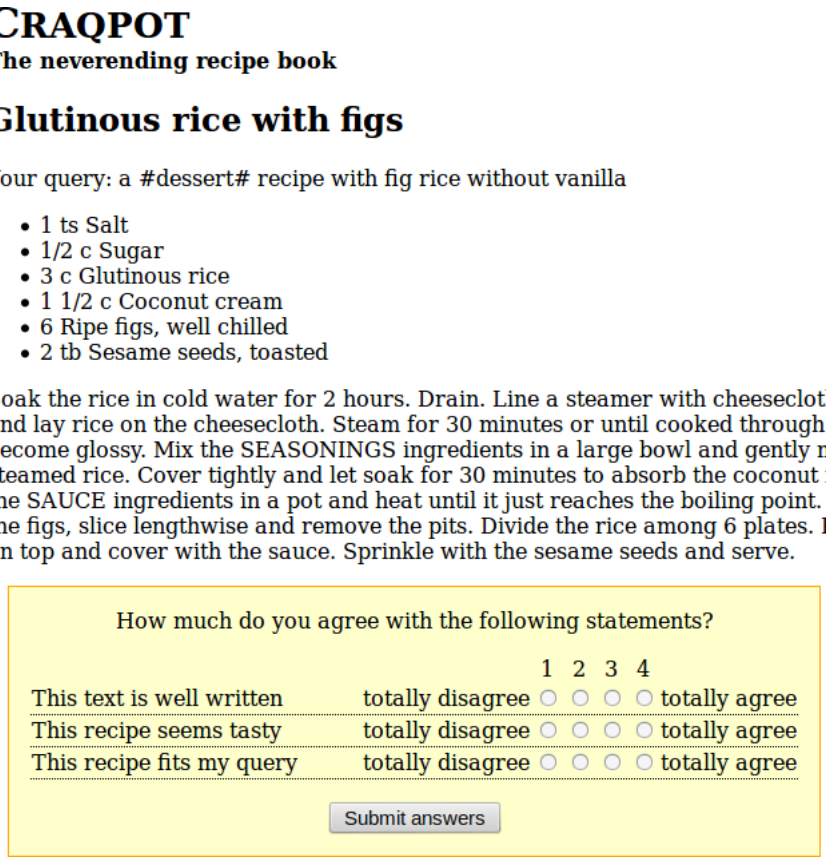


Figure 7.2 – Craqpot response and evaluation interface. This screenshot shows a recipe that was adapted by the CookIIS method.

	Retrieval	Craqpot	CookIIS
Text quality	.86	.77	.70
Recipe quality	.88	.72	.61
Fitness	.94	.91	.41

Table 7.1 – Average normalised user evaluations.

	Retrieval	Craqpot	CookIIS
Text quality	4	4	3
Recipe quality	4	3	3
Fitness	4	4	1

Table 7.2 – Modes of user evaluations.

function of the same. A high, narrow peak indicates uniformity among the results; the most right the peak, the better. While it is common to analyse observations obtained from a Likert scale as an interval variable, this practice is frowned upon by statisticians. For this reason, the modes are shown as well in table 7.2. Those are preliminary results based on only 50 evaluations, but they are very encouraging.

Wilcoxon signed-rank tests were performed for each criterion to compare methods pairwise and measure the probability that the observed differences in scores are the result of chance. The resulting p-values thresholds are shown in table 7.3. It is commonly assumed that p-values between .05 and .1 offer a weak presumption against H_0, whereas p-values below .05 offer a strong presumption against H_0. Values below .01 offer a very strong presumption.

In all three indicators, retrieval ranked first, and Craqpot ranked second. As expected, Craqpot's and CookIIS overall performance is worse than simple retrieval, partially validating H_1. This is mitigated, though, by the fact that simple retrieval was able to process only 46% of the queries assigned to it. In recipe quality, retrieval performed significantly better than Craqpot and strongly significantly better than CookIIS, but the score differ-

	Text quality		Recipe quality		Fitness	
	CookIIS	Craqpot	CookIIS	Craqpot	CookIIS	Craqpot
Retrieval	$< .05$	$< .1$	$< .01$	$< .05$	$< .001$	$> .5$
Craqpot	$> .5$		$< .5$		$< .001$	

Table 7.3 – Significance of the pairwise method comparisons.

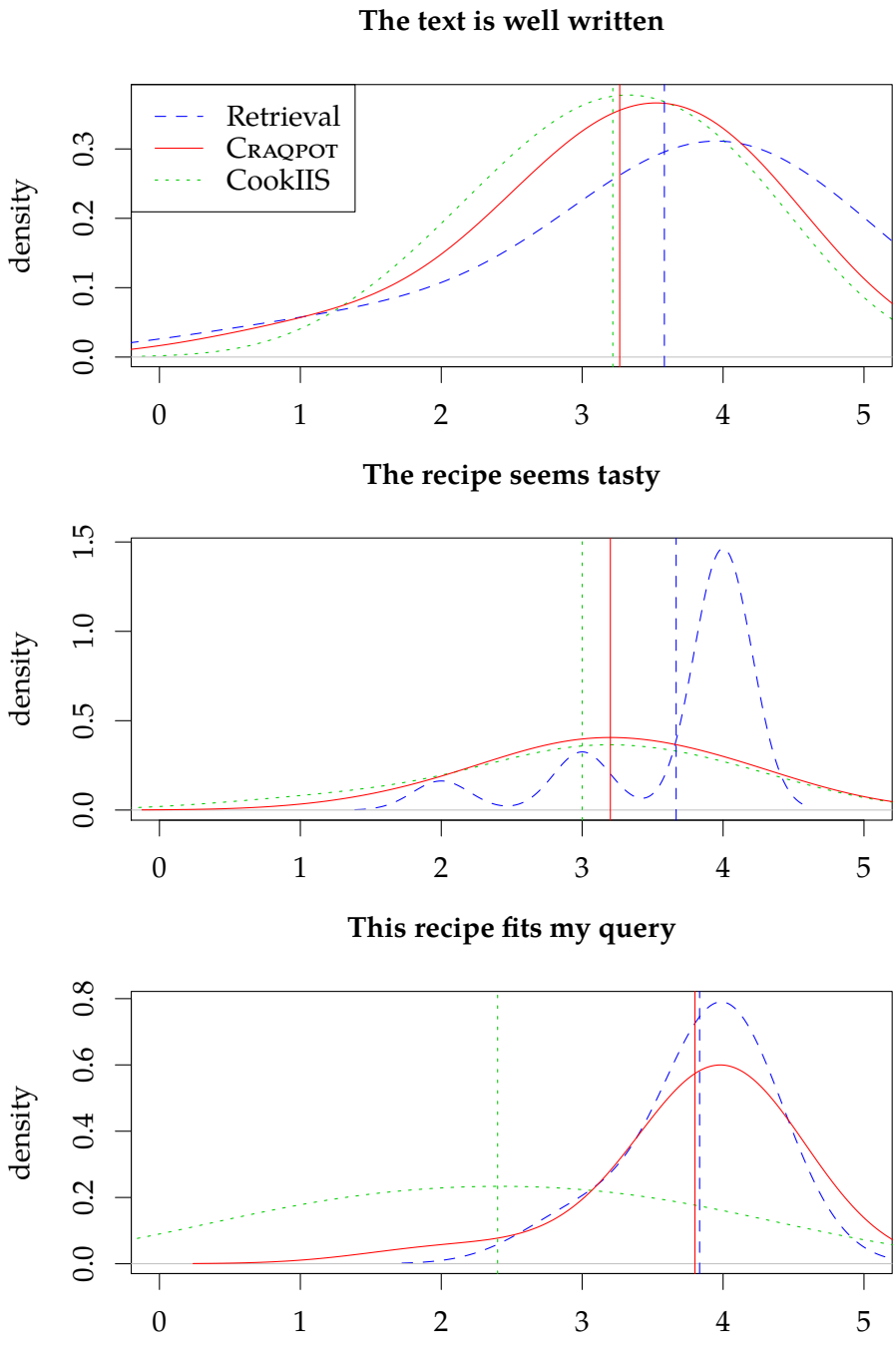


Figure 7.3 – Density distribution of user evaluations.

ence between Craqpot and CookIIS, while important, was not significant—the exact p-value is .30. This indicates that further tests would be necessary to decide on H_2. In text quality, retrieval performed strongly significantly better than Craqpot and very strongly significantly better than CookIIS. Although the evaluations surprisingly show that Craqpot did better than CookIIS, the difference is not statistically significant, confirming H_3. In fitness, Craqpot performed just as well as retrieval, and both methods were very strongly significantly better than CookIIS, confirming H_4.

The few available evaluations make it possible to claim that H_3 and H_4 are verified, and that H_1 is partially verified. More evaluations will be necessary to pronounce judgement on H_2.

7.2 Evaluation of Specific NLP Tasks

This section presents an evaluation for specific natural language processing tasks in the annotation process described in chapter 2. Those tasks are evaluated by creating a gold standard of texts that are assumed to be annotated correctly and comparing the automatic annotation obtained through our algorithms to this corpus. When the corpus is used to train a stochastic annotator, it is customary to use 90% of the tokens as training data and 10% as testing data. In our application, only the Brill part-of-speech tagger is concerned.

Assuming that the tokeniser works correctly, the first task to evaluate is the part-of-speech tagger. This task is evaluated with a single accuracy measure: the proportion of tokens that were tagged with the correct part-of-speech. Our tagger attains 93.1% accuracy, significantly under the 97.5% state-of-the-art claimed by Søgaard (2011). This low score can be explained by the small size of the training corpus used (12,125 tokens), whereas Brill taggers are routinely trained at least on the Brown corpus (Francis and Kučera, 1982), which has ca. 1,000,000 tokens. Unfortunately, because sentence structures in procedural texts tend to be different than those found in a general-purpose corpus, our accuracy actually decreases by 32% when we use a tagger trained on the Brown corpus.

The other annotation tasks are evaluated using precision and recall. Both measure the proportion of correct annotations with respect to, for precision, the total number of annotations added by the analysis process and, for recall, the total number of annotations found in the gold standard. In other words, a precision mistake can be seen as a false positive, and a recall mistake as a false negative. Table 7.4 shows the results for the annotation of the following features found in texts:

— Chunks.
— Clauses.
— Events of type “occurrence”, i.e. actions.

	Precision	Recall	$F_{0.5}$	Resolution
Chunks	.68	.87	.73	
Clauses	.68	.95	.75	
Prenitial occurences	1.00	.75	.90	
Occurences	.88	.78	.84	
States	1.00	.40	.67	
Simple references	1.00	.92	.97	.96
Complex references	1.00	.80	.92	.80
Zero-anaphoras	.67	.90	.73	.81
TIMEXes	1.00	1.00	1.00	
SIGNALs	1.00	.50	.75	

Table 7.4 – Actual accuracy of annotation tasks.

— Events of type “state”.
— Simple references to ingredients, i.e. using their name or a variant.
— Complex anaphoric references to ingredients of food components.
— Zero-anaphoric references to ingredients of food components.
— TimeML TIMEXes, i.e. explicit times and durations.
— TimeML SIGNALs, i.e. adverbial or adverbial-like modifiers.

$F_{0.5}$ is the weighted harmonic mean measure

$$F_w = (1 + w) \cdot \frac{\text{precision} \cdot \text{recall}}{w \cdot \text{precision} + \text{recall}} \tag{7.1}$$

commonly used for applications in which precision is preferable to recall—i.e. having correct information is more important than having complete information—which is consistent with the cautious approach to adaptation we advocate.

Because of a cascading failure effect, an important part of the annotation error can be explained with corresponding part-of-speech errors. To make it possible to evaluate the theoretical accuracy of the annotation process, we have run the same tests using texts with corrected parts-of-speech. The results are presented in table 7.5.

The theoretical results show that the algorithms we defined for text annotations have a good performance, except for part-of-speech tagging, which negatively effects the following tasks. This indicates that the latter would be the most appropriate candidate on which to devote extra research effort: our choice of a Brill tagger was based on convenience but, for instance, Marques and Lopes (2001) have shown good results with smaller corpora using neural networks.

Certain tasks depend on resources that may be too limited. For instance, the lower recall for occurences, states ans signals is largely due to the lexicon

	Precision	Recall	$F_{0.5}$
Chunks	.90	.97	.92
Clauses	.97	1.00	.98
Prenitial occurences	1.00	.75	.90
Occurences	1.00	.83	.94
States	1.00	.60	.82
Simple references	1.00	1.00	1.00
Complex references	1.00	.80	.92
Zero-anaphoras	.81	1.00	.86
`TIMEX`es	1.00	1.00	1.00
`SIGNAL`s	1.00	.50	.75

Table 7.5 – Theoretical accuracy of annotation tasks, without part-of-speech tagging cascading failure.

we use having a small size. The same is true for the target sets used to resolve existential anaphoras, as defined in section 2.2, which contains information about the nine most frequent nouns that were identified. The same section has identified a possible future work that would make it possible to compute more target sets automatically.

7.3 Perspective Probabilistic Evaluation

Given the availability of a very large corpus of recipes, we propose the following perspective. Language models are used to create a probability distribution for the appearance of specific words in specific contexts. This is used to compute the Shannon entropy of utterances. For instance, a language model trained on sufficiently many recipe texts would be likely to conclude that the sentence "Cream the butter" has less entropy than the sentence "Butter the cream".

We propose that entropy may be used as a measure of quality in textual adaptation. Our expectation is that the users of the system, given a recipe and two texts describing this recipe, may rate the text which has the lower entropy higher on the scale of text quality.

A similar idea could be applied with the case representation: a unigram probability could be assigned to each variable according to the frequency of the functor–parameter association (event–ingredient in the cooking domain) in a corpus, and aggregated with a bigram probability assigned to relations between variables. This would not be very different from a proposal to measure the confidence in workflow adaptation that has been tested and validated by Minor et al. (2012).

This approach would make it possible, given a set of test queries, to make

large scale tests rapidly, greatly aiding development. Under this framework, human users could still provide answers to to text and recipe quality questions, thereby providing a meta-evaluation.

Conclusion

This thesis has proposed a practical model making it possible to implement a case-based reasoning system to adapt procedures in case a defined precondition is not met, and to provide that adapted procedure to the user using natural language.

We argued that using natural language processing (NLP) was efficient and justified in this context. A framework and algorithms were provided that make it possible to annotate temporal information in procedural texts and to apply a regeneration process to the text when temporal information is to be modified.

Our main contributions in this area were:

— To provide a model for the resolution of evolutive anaphoras, i.e. words (or the absence of words) in a text that are substituted for objects that exist only at a specific moment in time, specifically for procedural texts.

— To provide a set of rules to translate certain linguistic features into variables and constraints expressed using a qualitative temporal algebra.

— To provide a general-purpose framework that make it possible to modify (regenerate) a text that has been annotated with our system to reflect modified temporal constraints.

Those constributions were implemented in a software library, which at this time is domain-specific. Making this library more relevant by separating the annotation/regeneration logic and the rules remains as a future work, as does a more rigourous study of our proposed regeneration framework.

Given a network of temporal qualitative constraints about actions expressed using a temporal algebra, we then showed that they could be adapted in a case-based reasoning (CBR) sense by relying on belief revision-based adaptation. To make this possible, it was necessary to formalise the notion of replacement through successive abstraction and refinement in qualitative constraint networks, and to develop a belief revision operator for qualitative algebras. We have also shown that this approach can be applied to different kinds of problems that can be formalised using temporal or spatial qualitative algebras, by providing an agricultural example.

Our main contributions in this area were:

— To provide and implement a belief revision operator for qualitative algebras that is compliant with Alchourrón, Gärdenfors, and Makinson (AGM) postulates. This operator, based on existing work on belief merging, uses an A* algorithm to search the model space.
— To provide and implement a belief revision operator for qualitative algebras, that is not compliant with all the AGM postulates, but which is tractable.

The former revision operator was publically released as a part of Revisor, a collection of revision and CBR revision-based adaptation engines, although some possible optimisations have been identified. Studying the properties of the latter revision operator remains as a future work.

Overall, we have offered a new model for adaptation in process-oriented case-based reasoning which reuses existing and well-studied formalisms, complete with inference procedures that can be fruitfully exploited. Using those formalisms, along with NLP annotations, make it possible for what we believe to be the first time to design a textual CBR system that is able to perform text adaptation.

Finally, we have assembled the implementations for the different parts of our work, added a module providing domain knowledge using our annotation process, and provided a web interface. We named this web application Craqpot. This has made it possible to evaluate our proposal and compare it to a system that uses retrieval and not adaptation, giving results of great quality to 46% of the queries but no result at all to the others, and to a system that performs minimal, string-substitution based adaptation.

Preliminary results based on 9 users evaluating the different system on 50 queries suggest that Craqpot is slightly inferior to the retrieval-only system on text and recipe quality, but superior to the other systems on those same criteria. They also show that users think that Craqpot's answers fit their queries just as well as simple retrieval, and much better than the other system.

Therefore, it is safe to assume that Roseline's husband will be able to enjoy a tasty carrot risotto that was cooked following a well-written recipe tonight.

Appendix A

Annotation rules

This appendix lists the nine annotation rules defined in and used by CRAQPOT, as described in section 4.2.

Annotation rule 1. *Given:*
- — e_1 *and* e_2 *are culinary events*
- — e_1 *and* e_2 *are realised by verbs is the same tense and mood*
- — e_2 *has a modifier which is an empty or "then" signal*
- — *the clauses of* e_1 *and* e_2 *are in the same sentence and the former immediately precedes the latter* or *the clause of* e_1 *is the last clause of sentence* s_1 *and the clause of* e_2 *is the first clause of sentence* s_2 *and* s_1 *immediately precedes* s_2

Annotate: 'e_1' $\{b,m\}^?$ 'e_2'

Annotation rule 2. *Given:*
- — e_1 *and* e_2 *are culinary events*
- — e_1 *is realised by a present imperative verb*
- — e_2 *is realised by a present participle verb*
- — *the clauses of* e_1 *and* e_2 *are in the same sentence*
- — *there is no other clause in this sentence* or *the clause of* e_2 *immediately follows the clause of* e_1
- — *there is no signal specifying a relation between* e_1 *and* e_2

Annotate: 'e_1' $\{di\}^?$ 'e_2'

Annotation rule 3. *Given:*
- — e_1, e_2 *are culinary events*
- — e_1 *and* e_2 *are realised by verbs in the same tense and mood*
- — e_2 *has a modifier which is a "meanwhile" signal*
- — *the clauses of* e_1 *and* e_2 *are in the same sentence and the former immediately precedes the latter or the clause of* e_1 *is the last clause of sentence* s_1 *and the clause of* e_2 *is the first clause of sentence* s_2 *and* s_1 *immediately precedes* s_2

Annotate: 'e_1' $\{di,fi,o\}^?$ 'e_2'

Annotation rule 4. *Given:*
- — e_1, e_2, e_3 *are culinary events*
- — e_1 *and* e_2 *are realised by verbs in the same tense and mood*
- — e_2 *has a modifier which is a "meanwhile" signal*
- — *the clauses of* e_1 *and* e_2 *are in the same sentence and the former immediately precedes the latter or the clause of* e_1 *is the last clause of sentence* s_1 *and the clause of* e_2 *is the first clause of sentence* s_2 *and* s_1 *immediately precedes* s_2
- — e_3 *has a member of the output of* e_1 *as its input*

Annotate: 'e_1' $\{b,m\}^?$ 'e_3'

Annotation rule 5. *Given:*
- — e_1, e_2 *are culinary events*
- — e_2 *is part of an "until" signal of* e_1
- — e_2 *is realised by an adjective*

Annotate: 'e_1' $\{m\}^?$ 'e_2'

Annotation rule 6. *Given:*
- — e_1, e_2 *are culinary events*
- — *the output of* e_1 *is the input of* e_2
- — e_1 *occurs in the ingredient list*

Annotate: 'e_1' $\{b\}^?$ 'e_2'

Annotation rule 7. *Given:*
- — e_1, e_2 *are culinary events*
- — t_1 *is a timex*
- — e_1 *and* e_2 *are realised by verbs in he same tense and mood*
- — t_1 *occurs inside an "after" signal*
- — e_2 *has a modifier which is the "after" signal containing* t_1
- — *the clauses of* e_1 *and* e_2 *are in the same sentence and the former immediately precedes the latter or the clause of* e_1 *is the last clause of sentence* s_1 *and the clause of* e_2 *is the first clause of sentence* s_2 *and* s_1 *immediately precedes* s_2

Annotate: 'e_1' $\{m\}^?$ '$supp_i$' $\{m\}^?$ 'e_2', '$supp_i$' $?^=$ 't_1'

Annotation rule 8. *Given:*
- — e_1, e_2 *are culinary events*
- — t_1 *is a timex*
- — e_1 *and* e_2 *are realised by verbs in he same tense and mood*
- — t_1 *occurs inside an "before the end" signal*
- — e_2 *has a modifier which is the "before the end" signal containing* t_1
- — *the clauses of* e_1 *and* e_2 *are in the same sentence and the former immediately precedes the latter or the clause of* e_1 *is the last clause of sentence* s_1 *and the clause of* e_2 *is the first clause of sentence* s_2 *and* s_1 *immediately precedes* s_2

Annotate: 'e_1' $\{fi^<\}$ '$supp_i$' $\{si^>,eq^=\}$ 'e_2', '$supp_i$' $?^=$ 't_1'

Annotation rule 9. *Given:*

— e_1 *is a culinary event*

— t_1 *is a timex*

— t_1 *occurs inside an "for" signal*

— e_1 *has a modifier which is the "for" signal containing* t_1

Annotate: 'e_1' ?= 't_1'

Appendix B

Regular Phrase Structure Grammar

This appendix lists the phrase structure grammar implemented by the chunker described in subsection 2.1.3. The categories part of speech tags used are Brown's.

— verb → VB
— verbg → VBG
— noun → NN | NNS
— prep → IN | TO
— pfet → ABL | ABN | ABX | AP
— det → AT | WDT | CD
— adj → JJ | ABN
— part → RP
— conj → CC | CC-TL
— comm → ,
— subc → CS
— adv → RB
— stop → .
— punct → comm | stop
— adjp → QL? adj QLP?
— VP → verb part?
— VGP → verbg part?
— NP → pdet? det? adjp* noun+
— NP → NP subc NP punct
— PP → prep adv* NP
— AP → subc adv* adjp+ adv*
— AP → stop adv comm

Bibliography

ISO/IEC 19501:2005: Information technology – Open distributed processing – Unified Modeling Language (UML) version 1.4.2. International Organization for Standardization, Geneva, Switzerland, 2005.

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

Ibrahim Adeyanju, Nirmalie Wiratunga, Robert Lothian, Somayajulu Sripada, and Luc Lamontagne. Case retrieval reuse net (CR2N): An architecture for reuse of textual solutions. In Lorraine McGinty and David C. Wilson, editors, *Case-Based Reasoning Research and Development*, volume 5650 of *Lecture Notes in Computer Science*, pages 14–28. Springer, 2009.

Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

James F. Allen. An interval-based representation of temporal knowledge. In *International Joint Conference on Artificial Intelligence*, pages 221–226, 1981.

James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

Douglas E. Appelt. Bidirectional grammars and the design of natural language generation systems. In *Proceedings of the 1987 workshop on Theoretical issues in natural language processing*, pages 206–212. Association for Computational Linguistics, 1987.

Fadi Badra, Julien Cojan, Amélie Cordier, Jean Lieber, Thomas Meilender, Alain Mille, Pascal Molli, Emmanuel Nauer, Amedeo Napoli, Hala Skaf-Molli, and Yannick Toussaint. Knowledge acquisition and discovery for the textual case-based cooking system WikiTaaable. In Sarah Jane Delany, editor, *ICCBR 2009 Workshop Proceedings*, pages 249–258, July 2009.

Philippe Balbiani and Aomar Osmani. A model for reasoning about topologic relations between cyclic intervals. In *Knowledge Representation*, pages 378–385, 2000.

Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. A model for reasoning about bidimensional temporal relations. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference*, pages 124–130, 1998.

John A. Bateman. Sentence generation and systemic grammar: an introduction. *Iwanami Lecture Series: Language Sciences, Iwanami Shoten Publishers, Tokyo*, 1997.

Brandon Bennett. Spatial reasoning with propositional logics. In *Knowledge Representation*, pages 51–62, 1994.

Ralph Bergmann and Yolanda Gil. Retrieval of semantic workflows with knowledge intensive similarity measures. In Ashwin Ram and Nirmalie Wiratunga, editors, *Case-Based Reasoning Research and Development*, volume 6880 of *Lecture Notes in Computer Science*, pages 17–31. Springer Berlin Heidelberg, 2011.

Ralph Bergmann and Alexander Stromer. MAC/FAC retrieval of semantic workflows. In *The Twenty-Sixth International FLAIRS Conference*, 2013.

Jenia Berzak, Dieu-Thu Le, and Ripan Hermawan. Natural language processing of recipes. Tutored project report, June 2009.

Christian Bessière. A simple way to improve path consistency processing in interval algebra networks. In *Proceedings of the 13th national conference for artificial intelligence (AAAI 1996)*, pages 375–380, 1996.

Steven Bird. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL conference on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

Giampio Bracchi and Barbara Pernici. The design requirements of office systems. *ACM Transactions on Information Systems*, 2(2):151–170, May 1984.

Eric Brill. A simple rule-based part of speech tagger. In *Workshop on Speech and Natural Language*, pages 112–116. Association for Computational Linguistics, 1992.

Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14(6):1007–1018, 2003.

Jaime G. Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. In Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell, editors, *Machine Learning*, volume 1 of *Symbolic Computation*, pages 137–161. Springer Berlin Heidelberg, 1983.

Jaime G. Carbonell. Derivational analogy : a theory of reconstructive problem solving and expertise acquisition. Technical Report 1534, Carnegie Mellon University, 1985.

Bowman L. Clarke. A calculus of individuals based on connection. *Notre Dame Journal of Formal Logic Notre-Dame*, 22(3):204–219, 1981.

John C. Clifton-Brown, Paul F. Stampfl, and Michael B. Jones. Miscanthus biomass production for energy in europe and its potential contribution to decreasing fossil fuel carbon emissions. *Global Change Biology*, 10(4): 509–518, 2004.

Julien Cojan and Jean Lieber. Conservative adaptation in metric spaces. In Klaus-Dieter Althoff, Ralph Bergmann, Mirjam Minor, and Alexandre Hanft, editors, *Advances in Case-Based Reasoning*, volume 5239 of *Lecture Notes in Computer Science*, pages 135–149. Springer Berlin Heidelberg, 2008.

Julien Cojan, Valmi Dufour-Lussier, Alice Hermann, Florence Le Ber, Jean Lieber, Emmanuel Nauer, and Gabin Personeni. Révisor : un ensemble de moteurs d'adaptation de cas par révision des croyances. In Sébastien Konieczny and Nicolas Maudet, editors, *Septièmes Journées d'Intelligence Artificielle Fondamentale*, pages 77–86, 2013.

Jean-François Condotta, Souhila Kaci, and Nicolas Schwind. A framework for merging qualitative constraints networks. In *FLAIRS Conference*, pages 586–591, 2008.

Jean-François Condotta, Souhila Kaci, Pierre Marquis, and Nicolas Schwind. Merging qualitative constraint networks defined on different qualitative formalisms. In KathleenStewart Hornsby, Christophe Claramunt, Michel Denis, and Gérard Ligozat, editors, *Spatial Information Theory*, volume 5756 of *Lecture Notes in Computer Science*, pages 106–123. Springer Berlin Heidelberg, 2009.

Amélie Cordier, Valmi Dufour-Lussier, Jean Lieber, Emmanuel Nauer, Fadi Badra, Julien Cojan, Emmanuelle Gaillard, Laura Infante-Blanco, Pascal Molli, Amedeo Napoli, and Hala Skaf-Molli. *Successful Case-based Reasoning Applications*, chapter TAAABLE: a Case-Based System for personalized Cooking. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2013. In press.

Mukesh Dalal. Investigations into a theory of knowledge base revision: preliminary report. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 475–479, 1988.

Dominique D'Almeida, Mouny Samy Modeliar, and Nicolas Schwind. On the neighborhood and distance between qualitative spatio-temporal configurations. In *International Workshop on Spatio-Temporal Dynamics*, pages 53–60, 2012.

Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1–3):61–95, 1991.

Michel Dojat, Nicolas Ramaux, and Dominique Fontaine. Scenario recognition for temporal reasoning in medical domains. *Artificial Intelligence in Medicine*, 14(1–2):139–155, 1998. Selected Papers from AIME '97.

Valmi Dufour-Lussier, Jean Lieber, Emmanuel Nauer, and Yannick Toussaint. Text adaptation using formal concept analysis. In Isabelle Bichindaritz and Stefania Montani, editors, *Case-Based Reasoning. Research and Development (ICCBR 2010)*, volume 6176 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2010.

Valmi Dufour-Lussier, Florence Le Ber, and Jean Lieber. Extension du formalisme des flux opérationnels par une algèbre temporelle. In Sébastien Konieczny, editor, *Sixièmes Journées de l'Intelligence Artificielle Fondamentale (IAF 2012)*, pages 133–142, Toulouse, France, May 2012a.

Valmi Dufour-Lussier, Florence Le Ber, Jean Lieber, and Laura Martin. Adapting spatial and temporal cases. In BelénDíaz Agudo and Ian Watson, editors, *Case-Based Reasoning Research and Development (ICCBR 2012)*, volume 7466 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2012b.

Valmi Dufour-Lussier, Florence Le Ber, Jean Lieber, and Laura Martin. Case adaptation with qualitative algebras. In Francesca Rossi, editor, *International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 3002–3006, Beijing, China, August 2013. AAAI.

Valmi Dufour-Lussier, Alice Hermann, Florence Le Ber, and Jean Lieber. Belief revision in the propositional closure of a qualitative algebra. In *14th International Conference on Principles of Knowledge Representation and Reasoning (KR 2014)*, Vienna, Austria, 2014a. AAAI Press.

Valmi Dufour-Lussier, Alice Hermann, Florence Le Ber, and Jean Lieber. Révision des croyances dans la clôture propositionnelle d'une algèbre qualitative. In *Journées d'intelligence artificielle fondamentale (IAF 2014)*, pages 117–128, Angers, France, 2014b.

Valmi Dufour-Lussier, Florence Le Ber, Jean Lieber, and Emmanuel Nauer. Automatic case acquisition from texts for process-oriented case-based reasoning. *Information Systems*, pages 153–167, mar 2014c.

Clarence A. Ellis and Gary J. Nutt. Office information systems and computer science. *ACM Computing Surveys*, 12(1):27–60, March 1980.

Winthrop Nelson Francis and Henry Kučera. *Frequency Analysis of English Usage*. Houghton Mifflin Company, 1982.

Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1–2):199–227, 1992.

Andrea Freßmann, Kerstin Maximini, Rainer Maximini, and Thomas Sauer. CBR-based execution and planning support for collaborative workflows. In Stefanie Brüninghaus, editor, *6th International Conference on Case-Based Reasoning Workshop Proceedings*, pages 271–280. Springer, 2005.

Jeffrey Friedl. *Mastering regular expressions*. O'Reilly, 2006.

Christian Fuchs, Christoph Gimmler, Simon Günther, Lukas Holthof, and Ralph Bergmann. Cooking cake. In *ICCBR 2009 Workshop Proceedings*, volume 9, pages 259–268, 2009.

Emmanuelle Gaillard, Laura Infante-Blanco, Jean Lieber, and Emmanuel Nauer. Tuuurbine: A generic CBR engine over RDFS. In *To appear in: International Conference for Case-Based Reasoning*, 2014.

Zeno Gantner, Matthias Westphal, and Stefan Wölfl. GQR: A fast reasoner for binary qualitative constraint calculi. In *AAAI'08 Workshop on Spatial and Temporal Reasoning*, volume 8, 2008.

Peter Gärdenfors. *Knowledge in flux: Modeling the dynamics of epistemic states.* MIT Press, 1988.

Peter Gärdenfors. *Belief Revision: An introduction*, chapter 1. Cambridge University Press, 1992.

Pablo Gervás, Raquel Hervás, and Juan Antonio Recio-García. The role of natural language generation during adaptation in textual CBR. In *4th Workshop on Textual Case-Based Reasoning: Beyond Retrieval (ICCBR 2007)*, pages 227–235, 2007.

Adam Grove. Two modellings for theory change. *Journal of Philosophical Logic*, 17(2):157–170, 1988.

William L. Harper. Rational conceptual change. In *Proceedings of the Biennial Meeting of the Philosophy of Science Association*, pages 462–494, 1976.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Jonathan Hillier, Carly Whittaker, Gordon Dailey, Matthew Aylott, Eric Casella, Goetz M Richter, Andrew Riche, Richard Murphy, Gail Taylor, and Pete Smith. Greenhouse gas emissions from four bioenergy crops in england and wales: integrating spatial estimates of yield and soil carbon balance in life cycle analyses. *GCB Bioenergy*, 1(4):267–281, 2009.

Julien Hué and Matthias Westphal. Revising qualitative constraint networks: Definition and implementation. In *Tools with Artificial Intelligence (ICTAI)*, pages 548–555, 2012.

Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. Lexical selection and paraphrase in a meaning-text generation model. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, volume 119 of *The Kluwer International Series in Engineering and Computer Science*, pages 293–312. Springer, 1991.

Michel Jaczynski. A framework for the management of past experiences with time-extended situations. In *Proceedings of the sixth international conference on Information and knowledge management*, pages 32–39. ACM, 1997.

Martha Dørum Jære, Agnar Aamodt, and Pål Skalle. Representing temporal knowledge for case-based prediction. In Susan Craw and Alun Preece, editors, *Advances in Case-Based Reasoning*, volume 2416 of *Lecture Notes in Computer Science*, pages 174–188. Springer Berlin Heidelberg, 2002.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2009.

Stelios Kapetanakis, Miltos Petridis, Brian Knight, Jixin Ma, and Liz Bacon. A case based reasoning approach for the monitoring of business workflows. In Isabelle Bichindaritz and Stefania Montani, editors, *Case-Based Reasoning Research and Development*, volume 6176 of *Lecture Notes in Computer Science*, pages 390–405. Springer Berlin Heidelberg, 2010.

Angela Karp and Goetz M. Richter. Meeting the challenge of food and energy security. *Journal of Experimental Botany*, 62(10):3263–3271, 2011.

Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, 1991.

Martin Kay. Syntactic processing and functional sentence perspective. In *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 12–15. Association for Computational Linguistics, 1975.

Stephen Cole Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata studies*. Princeton University Press, 1956.

Leila Kosseim, Agnès Tutin, Richard Kittredge, and Guy Lapalme. Generating grammatical and lexical anaphora in assembly instructional texts. In Giovanni Adorni and Michael Zock, editors, *Trends in Natural Language Generation An Artificial Intelligence Perspective*, volume 1036 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 1996.

Peter B. Ladkin. Time representation: A taxonomy of internal relations. In *AAAI*, pages 360–366, 1986.

Luc Lamontagne and Guy Lapalme. Textual reuse for email response. In Peter Funk and Pedro A. González Calero, editors, *Advances in Case-Based Reasoning*, volume 3155 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2004.

Isaac Levi. Subjunctives, dispositions and chances. *Synthese*, 34(4):423–455, 1977.

Jean Lieber. Strong, fuzzy and smooth hierarchical classification for case-based problem solving. In Frank van Harmelen, editor, *15th European Conference on Artificial Intelligence (ECAI'02)*, pages 81–85, Lyon, France, July 2002. IOS Press, Amsterdam.

Jean Lieber. Application of the revision theory to adaptation in case-based reasoning: The conservative adaptation. In Rosina O Weber and Michael M Richter, editors, *Case-Based Reasoning Research and Development*, volume 4626 of *Lecture Notes in Computer Science*, pages 239–253. Springer Berlin Heidelberg, 2007.

Gérard Ligozat. On generalized interval calculi. In *AAAI*, pages 234–240, 1991.

Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

Jixin Ma and Brian Knight. A framework for historical case-based reasoning. In Kevin D Ashley and Derek G Bridge, editors, *Case-Based Reasoning Research and Development*, volume 2689 of *Lecture Notes in Computer Science*, pages 246–260. Springer Berlin Heidelberg, 2003.

Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

Nuno C. Marques and Gabriel Pereira Lopes. Tagging with small training corpora. In *Advances in Intelligent Data Analysis*, pages 63–72. Springer, 2001.

Laura Martin, Julie Wohlfahrt, Florence Le Ber, and Marc Benoît. L'insertion territoriale des cultures biomasses pérennes. étude de cas sur le miscanthus en côte-d'or (bourgogne, france). *L'Espace géographique*, (2):138–153, 2012.

Christian Matthiessen. Lexico(grammatical) choice in text generation. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, volume 119 of *The Kluwer International Series in Engineering and Computer Science*, pages 249–292. Springer, 1991.

John McCarthy. Situations, actions, and causal laws. Technical report, Stanford University, 1963.

Mirjam Minor, Daniel Schmalen, and Ralph Bergmann. XML-based representation of agile workflows. In Martin Bichler, Thomas Hess, Helmut Krcmar, Ulrike Lechner, Florian Matthes, Arnold Picot, Benjamin Speitkamp, and Petra Wolf, editors, *Multikonferenz Wirtschaftsinformatik*, pages 439–440. GITO-Verlag Berlin, 2008.

Mirjam Minor, Ralph Bergmann, Sebastian Görg, and Kirstin Walter. Towards case-based adaptation of workflows. In Isabelle Bichindaritz and Stefania Montani, editors, *Case-Based Reasoning. Research and Development*, volume 6176 of *Lecture Notes in Computer Science*, pages 421–435. Springer Berlin Heidelberg, 2010.

Mirjam Minor, Mohd.Siblee Islam, and Pol Schumacher. Confidence in workflow adaptation. In BelénDíaz Agudo and Ian Watson, editors, *Case-Based Reasoning Research and Development*, volume 7466 of *Lecture Notes in Computer Science*, pages 255–268. Springer Berlin Heidelberg, 2012.

Stefania Montani and Giorgio Leonardi. A case-based approach to business process monitoring. In Max Bramer, editor, *Artificial Intelligence in Theory and Practice III*, volume 331 of *IFIP Advances in Information and Communication Technology*, pages 101–110. Springer Berlin Heidelberg, 2010.

Stefania Montani and Luigi Portinale. Accounting for the temporal dimension in case-based retrieval: A framework for medical applications. *Computational Intelligence*, 22(3-4):208–223, 2006.

Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. *Journal of the ACM*, 42(1):43–66, January 1995.

Régis Newo, Kerstin Bach, Alexandre Hanft, and Klaus-Dieter Althoff. On-demand recipe processing based on CBR. In *ICCBR 2010 Workshop Proceedings*, pages 209–218, 2010.

Louis Pillemer, Livia Blum, Irwin H. Lepow, Leona Wurz, and Earl W. Todd. The properdin system and immunity: III. The zymosan assay of properdin. *The Journal of experimental medicine*, 103(1):1–13, 1956.

Arun K. Pujari, G. Vijaya Kumari, and Abdul Sattar. INDU: An interval & duration network. In Norman Foo, editor, *Advanced Topics in Artificial Intelligence*, volume 1747 of *Lecture Notes in Computer Science*, pages 291–303. Springer Berlin Heidelberg, 1999.

James Pustejovsky, José M. Castano, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. TimeML: Robust specification of event and temporal expressions in text. *New directions in question answering*, 3:28–34, 2003.

Dragomir R. Radev. *Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources*. PhD thesis, Department of Computer Science, Columbia University, New York, April 1999.

Dragomir R. Radev and Kathleen R. McKeown. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):470–500, September 1998.

David A. Randell and Anthony G. Cohn. Modelling topological and metrical properties in physical processes. In R. J. Brachman, H. Levesque, and R. Reiter, editors, *Principles of Knowledge Representation and Reasoning*, pages 357–368, 1989.

David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *Knowledge Representation*, pages 165–176, 1992.

Raymond Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation*, 27:359–380, 1991.

Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning*. Psychology Press, 1989.

Karin Kipper Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, 2005.

Pol Schumacher, Mirjam Minor, and Erik Schulte-Zurhausen. On the use of anaphora resolution for workflow extraction. In Thouraya Bouabana-Tebibel and Stuart H. Rubin, editors, *Integration of Reusable Systems*, volume 263 of *Advances in Intelligent Systems and Computing*, pages 151–170. Springer International Publishing, 2014.

Nicolas Schwind. *Fusion de réseaux de contraintes qualitatives*. PhD thesis, Université d'Artois, Centre de Recherche en Informatique de Lens, Université d'Artois, Lens, France, 2010.

Andrea Setzer, Robert Gaizauskas, and Mark Hepple. Using semantic inferences for temporal annotation comparison. In *Fourth international workshop on inference in computational semantics (ICOS-4)*, pages 195–196, 2003.

Stuart M. Shieber. A uniform architecture for parsing and generation. In *Proceedings of the 12th conference on Computational linguistics*, pages 614–619. Association for Computational Linguistics, 1988.

Advaith Siddharthan. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109, 2006.

Advaith Siddharthan. Text simplification using typed dependencies: A comparison of the robustness of different generation strategies. In *13th European Workshop on Natural Language Generation*, pages 2–11. Association for Computational Linguistics, 2011.

Barry Smyth and Mark T. Keane. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102(2):249–293, 1998.

Anders Søgaard. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *ACL (Short Papers)*, pages 48–52, 2011.

Miquel Sánchez-Marré, Ulises Cortés, Montse Martínez, Joaquim Comas, and Ignasi Rodríguez-Roda. An approach for temporal case-based reasoning: Episode-based reasoning. In Héctor Muñoz-Ávila and Francesco Ricci, editors, *Case-Based Reasoning Research and Development*, volume 3620 of *Lecture Notes in Computer Science*, pages 465–476. Springer Berlin Heidelberg, 2005.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, pages 173–180. Association for Computational Linguistics, 2003.

Agnès Tutin and Richard Kittredge. Lexical choice in context: generating procedural texts. In *Proceedings of the 14th conference on Computational linguistics*, pages 763–769. Association for Computational Linguistics, 1992.

Josep Valls and Santiago Ontañón. Natural language generation through case-based text modification. In Belén Díaz Agudo and Ian Watson, editors, *Case-Based Reasoning Research and Development*, volume 7466 of *Lecture Notes in Computer Science*, pages 443–457. Springer, 2012.

Peter van Beek. Reasoning about qualitative temporal information. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 728–734, Boston, Massachusetts, July 1990.

Peter van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.

Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartosz Kiepuszewski, and Ana P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

Manuela Veloso, Hector Muñoz-Avila, and Ralph Bergmann. Case-based planning: selected methods and systems. *Artificial Intelligence Communications*, 9(3):128–137, 1996.

Marc Verhagen. Temporal closure in an annotation environment. *Language Resources and Evaluation*, 39(2–3):211–241, 2005.

Marc B. Vilain and Henry A. Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI*, volume 86, pages 377–382, 1986.

Kirstin Walter, Mirjam Minor, and Ralph Bergmann. Workflow extraction from cooking recipes. In *Proceedings of the ICCBR 2011 Workshops*, pages 207–216, 2011.

Christoph Wargitsch, Thorsten Wewers, and Felix Theisinger. WorkBrain: Merging organizational memory and workflow management systems. In *Workshop of Knowledge-Based Systems for Knowledge Management in Enterprises*, 1997.

Barbara Weber and Werner Wild. Towards the agile management of business processes. In Klaus-Dieter Althoff, Andreas Dengel, Ralph Bergmann, Markus Nick, and Thomas Roth-Berghofer, editors, *Professional Knowledge Management*, volume 3782 of *Lecture Notes in Computer Science*, pages 409–419. Springer Berlin Heidelberg, 2005.

Hélène W. Zub and Maryse Brancourt-Hulmel. Agronomic and physiological performances of different species of miscanthus, a major energy crop. a review. *Agronomy for Sustainable Development*, 30(2):201–214, 2010.

Abstract

This thesis proposes a practical model making it possible to implement a case-based reasoning system that adapts processes represented as natural language text in response to user queries. While the cases and the solutions are in textual form, the adaptation itself is performed on networks of temporal constraints expressed with a qualitative algebra, using a belief revision operator. Natural language processing methods are used to acquire case representations and to regenerate text based on the adaptation result.

Résumé

Cette thèse propose un modèle permettant la mise en œuvre d'un système de raisonnement à partir de cas capable d'adapter des procédures représentées sous forme de texte en langue naturelle, en réponse à des requêtes d'utilisateurs. Bien que les cas et les solutions soient sous forme textuelle, l'adaptation elle-même est d'abord appliquée à un réseau de contraintes temporelles exprimées à l'aide d'une algèbre qualitative, grâce à l'utilisation d'un opérateur de révision des croyances. Des méthodes de traitement automatique des langues sont utilisées pour acquérir les représentations algébriques des cas ainsi que pour regénérer le texte à partir du résultat de l'adaptation.