**UNIVERSITÉ PARIS 13 - SORBONNE PARIS CITÉ**

**LIPN - UMR CNRS 7030**

# THESE

Pour l'obtention du grade de

**DOCTEUR DE L'UNIVERSITÉ PARIS 13**

**Discipline** : Informatique

Présentée par

**Abdoulaye GUISSÉ**

# Une plateforme d'aide à l'acquisition et à la maintenance des règles métier à partir de textes règlementaires

Sous la direction : **M. François Lévy**

**30 Janvier 2013**

**Jury**

**M. Patrick Saint-Dizier**, Directeur de Recherche CNRS, IRIT, *rapporteur*
**M. Jean Charlet**, Chargé de mission Recherche AP-HP, HDR, INSERM, *rapporteur*
**M. Aurelien Max**, Maître de conférences, LIMSI, *examinateur*
**M. Patrick Albert**, Leader IBM CAS France, *examinateur*
**Mme Adeline Nazarenko**, Professeur, LIPN, *examinatrice*
**M. François Lévy**, Professeur, LIPN, *directeur de thèse*

**À Oumoyam Chérie,**
**et à Halima, ma fifille adorée.**

# Remerciements

Mes remerciements vont à toutes les personnes qui m'ont aidé de près ou de loin tout au long de ces années de thèse, et à qui je dédie de tout cœur ce travail :

François Lévy, mon directeur de thèse, qui aura été un admirable professeur, et qui par son expérience et sa rigueur a beaucoup contribué à la maturation de mon esprit de chercheur. Je le remercie pour sa patience, sa disponibilité, ses critiques et conseils toujours pertinents. Et humainement, il a été d'une gentillesse qui m'aura beaucoup aidé à prendre conscience du défi et à croire que j'avais les capacités nécessaires pour y arriver. Il m'a beaucoup aidé à organiser mes idées et à rendre plus compréhensible et accessible mon travail. Je le remercie pour toutes ces longues nuits qu'il a passées pour m'aider à corriger le mémoire final.

Adeline Nazarenko, qui m'a offert l'opportunité d'intégrer l'équipe RCLN et qui a participé si constamment à la réflexion qui nourrit cette thèse. Ce fut un honneur pour moi de travailler avec elle et surtout de bénéficier de son expérience et sa connaissance du domaine et des questions qui m'intéressaient le plus. La collaboration avec elle m'aura été d'une grande utilité et mon amour pour la recherche en sort encore plus grandi. Je la remercie pour les idées constructives et toujours pertinentes qu'elle n'hésitait pas à me proposer pour m'aider à comprendre et à avancer.

Patrick Saint-Dizier et Jean Charlet, qui m'ont fait l'honneur d'être mes rapporteurs de thèse. Leurs deux relectures rigoureuses du mémoire ainsi que toutes leurs suggestions de corrections m'ont été très utiles pour clarifier et améliorer le message que je défends. J'en garderai un acquis au delà de cette soutenance.

Patrick Albert et Aurelien Max, pour l'intérêt qu'ils ont présenté pour cette thèse, Patrick Albert durant le projet ONTORULE, Aurélien Max en acceptant de participer à ma soutenance de mi-parcours et pour avoir bien volontiers accepté de faire partie de mon Jury.

Tous les membres du laboratoire LIPN et de l'équipe RCLN, notamment tous ceux qui ont été très aimables à mon égard et dont les discussions et conseils m'ont été très précieux. Je garde

en mémoire les discussions entre thésards qui ont facilité beaucoup de mes réflexions autour de diverses questions, le groupe Atelier état de l'art (avec Nouha, Sarra, jonathan, Sondes, Haifa et Sylvie Salotti), le groupe OntoRule Paris 13 (avec Nouha, François, Adeline et Syvie Szulman) et le cadre familial qui s'était installé dans notre bureau et sur notre pallier (au 3ième) avec mes collègues de bureau (Zaid, Yue, Manisha, Nesrine, Alois, Hanane).

Les partenaires du projet ONTORULE avec qui j'ai beaucoup discuté et interagi notamment les gens d'IBM Adil, Patrick, Amina, le groupe CTIC spécialiste des technologies du Web sémantique, et aussi John Hall un expert des règles métier et de SBVR.

Tous ces amis précieux avec qui je discute et partage de vraies valeurs notamment Gaoussou, Mouha, Déthié, Sowdam, Khady Thiam, laye Ndiaye, Daff, Babs (Nice), ainsi que mon guide Oustaze.

Ma famille, en particulier mes parents, ma tante, mes frères et sœurs qui ne cessent de me soutenir quel que soit ce que j'entreprends et tout au long de mon cursus scolaire.

Et enfin, ma petit famille à moi, mon épouse, Oumou khaïry, mon amie, mon amour et mon premier soutien dont la présence, la patience et la tendresse m'ont donné le courage d'y croire jusqu'au bout ; et ma fifille et princesse adorée Halima dont la venue aura été un bonheur immense et un grand déclic dans ma vie.

# Table des matières

# Résumé

Les systèmes décisionnels reposent généralement sur un ensemble de règles métier qui contraignent et conditionnent les données du système d'information d'une entreprise. Elles peuvent concerner aussi bien le calcul de l'impôt d'un contribuable que l'attribution de points de fidélité à des clients ou le contrôle de qualité d'un processus industriel.

Cette thèse a été conduite dans le cadre du projet européen ONTORULE (ONTOlogy meets business RULEs). Elle s'intéresse à l'acquisition des règles à partir des textes règlementaires originaux produits par des experts métier, de l'entreprise ou extérieurs. Notre objectif est d'aider les experts, qui comprennent le mieux ces textes, à traduire les règles du langage naturel vers un langage plus simple et moins ambigu. De plus, nous proposons d'aider au diagnostic des problèmes de maintenance causés par des incohérences produites tout au long de la transformation des règles ou lorsque les textes évoluent.

Ces règles sont généralement disponibles dans des documents rédigés dans une forme de langage naturel souvent marquée par leur valeur réglementaire. Afin de les traduire en règles exécutables et de les intégrer dans des systèmes d'aide à la décision, il est nécessaire d'identifier dans les textes les fragments textuels représentant des règles. Puis il faut conduire une transformation des règles en langage naturel en des règles formelles. Se pose alors le problème d'alléger la charge de cette transformation, d'autant plus lourde que la complexité du langage employé en réserve l'interprétation et la manipulation à des experts. Nous proposons d'outiller l'explicitation des règles initiales en une expression facilement compréhensible, à partir de laquelle peut commencer le travail de formalisation et d'implémentation.

Pour se faire, nous proposons aux experts métier un cadre interactif qui les aide à mieux structurer et organiser le travail d'explicitation. Ce cadre comporte :

– une méthodologie d'acquisition en deux étapes : la première consiste en une fouille à partir de la représentation syntaxique et sémantique des textes afin d'identifier les phrases représentant des règles ; et la seconde étape consiste à normaliser ces phrases via un langage naturel contrôlé comme SBVR SE afin de les simplifier, les désambiguïser et faciliter leur traduction dans un langage formel.
– un modèle sémantique documenté qui définit une structure de navigation pour assister l'acquisition et articuler les textes règlementaires, les concepts métier de domaine décrits par une ontologie et l'ensemble de règles extraites. Cette structure, sur la base de ses fonctionnalités de traçabilité, sert aussi pour la gestion des incohérences dans les règles et de l'évolution des textes. Le modèle est formellement représenté avec des standards du Web sémantique (RDF, RDFs, RDFa, OWL, SPARQL) à travers un graphe RDF.
– une plateforme, SemEx, qui fournit d'une part des interfaces pour la navigation et l'interrogation d'un graphe RDF en utilisant le moteur de raisonnement CORESE, d'autre part une interface d'édition de règles implémentant la méthodologie d'acquisition. SemEx est une application open source en Java.

Mots-clés : Règles métiers, textes règlementaires, Web sémantique, modélisation et acquisition des connaissances.

# Abstract

Decision support systems are usually based on a set of business rules that constrain the information system data of a company. They can cope with e.g. computing the amount of taxes that is owed by a taxpayer as well as awarding benefits to clients of a company or controling the quality of an industrial process.

This thesis has taken place in the framework of the European project OntoRule (ONTOlogies meet business RULEs). It focuses on the acquisition of rules from original regulatory texts written by business experts, belonging to the company or not. Our goal is to help experts, who best understant these texts, to translate rules from natural language toward a simpler and less ambiguous language. Moreover, we propose to help diagnosing maintenance problems caused by incoherencies happening either along the translation of rules, or when the original texts evolve.

These rules are generally available in documents written in natural language, and their regulatory character marks the form of this language. In order to translate them into executable rules and to integrate them in decision making systems, text fragments embeding rules must first be identified. Then a transformation must be driven from natural language rules to formal rules. The problem is then to reduce the burden of this transformation, at the more heavy since this complex language can only be safely handled and interpreted by experts. We propose to equip the task of explicitating initial rules into an easily understandable expression, so that, relying on this expression, formalization and implementation work can start.

In order to do so, we propose to business experts an interactive framework which helps them structuring and organizing better the explicitation work. This framework consists of :

– a two-steps acquisition methodology : a first step organizes a search in the syntactic and semantic representations of texts in order to identify sentences which are relevant as rules ;

and a second step consists in normalizing these sentences, through a controlled natural language as SBVR-SE, in order to simplify them, disambiguate them and make easier their translation into a formal language.

– A documented semantic model that defines a navigation structure to assist acquisition and articulate regulatory texts, business concepts described by an ontology and the set of extracted rules. This structure, relying on its tracing functionality, also supports the management of incoherency in rules relatively to needs as identified in the initial, and of changes when initial texts evolve. The model is formally represented according to Semantic Web technologies (RDF, RDFS, RDFa, OWL, SPARQL) through an RDF graph.
– a platform ; SemEx, which provides on the one hand interfaces to navigate and query an RDF graph with the help of the CORESE inference engine, on the other hand an interface to edit rules according to the acquisition methodology. SemEx is an open source application written in Java.

Keywords : Business rules, Regulations, Semantic web, Knowledge modeling, knowledge acquisition.

# Chapitre I

# Introduction

Cette thèse présente un travail d'ingénierie de la connaissance relatif à la mise en œuvre des règles métier. Elle porte plus précisément sur leur acquisition, leur formalisation et leur maintenance. La gestion des règles métier prend tout son sens dans le développement des Systèmes de Gestion des Règles Métier (SGRM), les descendants des systèmes experts ([Legendre et al., 2010]). Ces derniers sont apparus dans le domaine de l'intelligence artificielle comme des outils de reproduction des comportements, des raisonnements et des mécanismes cognitifs d'un expert humain dans une application particulière. Les SGRM sont utilisés pour prendre en charge les règles métier qui servent à formaliser les connaissances de l'expert ; ceci dans une optique d'automatisation de son savoir-faire et d'aide à la prise de décisions d'une entreprise ou de contrôle de celle-ci. Les réponses du SGRM peuvent concerner aussi bien le calcul de l'impôt d'un contribuable que l'attribution de points de fidélité à des clients ou le contrôle de qualité d'un processus industriel. Chaque règle métier correspond à une déclaration qui définit ou contraint certains aspects de l'entreprise ; elle est reprise pour contraindre et conditionner les données de son système d'information (SI).

## I.1 Contexte institutionnel

Cette thèse s'inscrit dans le cadre du projet européen ONTORULE [1] qui visait à faciliter l'interaction, la gestion et le contrôle des applications métier par les acteurs du domaine en séparant une couche conceptuelle, modélisant les connaissances métier, du code des applications. Cela suppose de développer les modèles formels, les méthodes et les outils permettant d'acquérir et de raisonner au niveau conceptuel. Comme le nom du projet l'indique, les connaissances métier dont il est question sont des ontologies de domaine et des règles métier. Il s'agit

1. Projet EU-IST Integrated Project 2009-231875 ONTORULE ("Ontologies meet business rules").

donc d'intégrer tous les éléments nécessaires d'une part à l'acquisition d'ontologies et de règles métier à partir des sources appropriées, y compris les textes réglementaires en langage naturel, et d'autre part à leur gestion, à leur maintenance et à leur utilisation transparente dans les applications métier.

Pour donner corps à cette visée, l'objectif d'ONTORULE était de concevoir des plates-formes d'acquisition, de maintenance et d'exécution des règles qui rendent aux gens du métier la main sur les connaissances métier, de manière à ce qu'ils puissent à la fois :

- acquérir l'ontologie et les règles métier à partir de textes,
- assurer leur mise à jour (en cas d'incohérence ou en cas d'évolution des textes) et comprendre le pourquoi des décisions proposées par le SGRM.

L'équipe RCLN du LIPN apportait à ce projet sa compétence en ingénierie des connaissances textuelles. Elle s'y est intéressée à la manière dont les textes réglementaires peuvent être pris en compte par les méthodes d'acquisition actuelles. Deux thèses ont alors été proposées, l'une sur l'acquisition de l'ontologie décrivant le vocabulaire métier et l'autre sur l'acquisition des règles. C'est cette dernière qui est traitée dans ce mémoire.

## I.2 Règles et textes

Ce travail se focalise donc essentiellement sur la prise en charge des règles métier exprimées dans des ressources textuelles et sur leur intégration dans les SGRM. Nous nous interrogeons sur la problématique de leur repérage dans ces textes et du travail de formalisation qui doit précéder leur automatisation. De ce fait, nous nous intéressons essentiellement à deux tâches :

(i) L'une des tâches principales de la mise au point du SGRM est celle de **l'acquisition des règles**. Pour une bonne part, ces règles sont disponibles initialement sous forme rédigée, décrites en langage naturel par les experts métier dans des textes réglementaires [2]. Ces derniers sont décrits comme étant la matière première de l'entreprise et sont utilisés comme sources d'informations pour alimenter son système d'information. Ces textes sont connus pour leur caractère juridique et la complexité de leurs tournures linguistiques. Ils sont souvent difficiles à décrypter si on n'est pas expert du domaine.

L'acquisition pose d'abord un problème d'identification des règles à partir des textes. Pour cela, des méthodes de fouilles textuelles (voir II.2.2.1) basées sur des techniques statis-

2. Nous appelons ici "texte réglementaire" tout texte qui décrit un comportement obligatoire dans l'entreprise ou l'institution. Au sens du droit français ça peut être une loi, un contrat, etc. Pour l'essentiel, la source alternative de règles métier est l'application de tables de décisions.

tiques ou syntaxiques ont été proposées dans l'état de l'art. Elles montrent des résultats assez satisfaisants pour détecter les fragments de texte descriptifs de règles métier valides, sous réserve que la complexité du discours dans les textes ne soit pas trop forte. Ensuite, une fois les règles en langage naturel (LN) identifiées, l'acquisition pose un problème de traduction de celles-ci vers une forme interprétable par la machine. à l'heure actuelle très peu de travaux abordent cette question, et ils montrent qu'il est difficile de traduire le langage naturel dans un langage formel (LF), encore plus d'automatiser ce passage. Nous préconisons un passage intermédiaire par les langages contrôlés (LC, voir III.1) pour faciliter cette traduction et décomposer le problème en deux parties, la traduction du langage naturel au langage contrôlé et celle du langage contrôlé au langage formel. D'un côté, les langages contrôlés sont déjà très contraignants du fait de leur expressivité limitée pour traduire le langages naturel et ses complexités. D'un autre côté, le passage du langage contrôlé au langage formel pose aussi problème du fait de la variabilité des langages de règles (il n'est pas clair qu'un langage formel cible commun puisse être défini), et parce que certaines structures linguistiques exprimables dans le langage contrôlé ne se transposent formellement que moyennant des ajouts substantiels de connaissances. Ce sont là des questions d'une nature très différente de celles qui se posent dans la transformation en langage contrôlé ; aussi cette thèse s'intéresse-t-elle uniquement au passage du langage nature au langage contrôlé.

(ii) La seconde tâche est la maintenance des règles. Elle va de pair avec l'acquisition car, il est nécessaire que les règles extraites soient cohérentes et conformes aux besoins du système d'information. Leur mise à jour s'impose alors, notamment lorsque des incohérences sont détectées dans les règles, ou lorsque les textes évoluent :

- Les incohérences montrent une non-conformité entre les textes et règles extraites – sauf si les textes étaient déjà incohérents. Cela veut dire que des erreurs ont sans doute été commises lors du passage du langage naturel au formel. Dans ce cas, la tâche est de diagnostiquer les incohérences et de corriger les règles affectées.
- Lorsque les textes évoluent, il est légitime que les règles extraites soient maintenues afin de garder une conformité entre les textes et les règles. Dans ce cas, la tâche est de détecter les changements et de mettre à jour les règles dérivées des textes changés, soit en les modifiant, soit en supprimant celles qui sont devenues obsolètes.

Des travaux de recherche ont exploré ces questions de maintenance mais sans prendre en compte les textes. En effet, dans la plupart des travaux, les textes sont utilisés seulement comme sources d'information et une fois les règles extraites, le lien est perdu. éventuellement, les règles sont retraduites en langage naturel ou contrôlé pour permettre aux experts

métier de corriger les incohérences, de modifier ou de supprimer celles qui évoluent. Cette thèse tente d'éviter la retraduction et réfléchir sur la manière de faire jouer un rôle aux textes dans la maintenance des règles.

## I.3 Cas d'usage

Nous avons testé et en partie construit nos idées sur des cas d'usages qui ont été proposés dans le projet OntoRule. Ces cas d'usages proviennent de contextes industriels réellement applicatifs et nous avons utilisé les textes réglementaires associés :

– Le cas d'usage ***AAdvantage*** vise une application de gestion du programme de fidélisation d'American Airlines. Il s'agit de déterminer les points de fidélité qu'un voyageur membre du programme a gagné sur une période donnée. Les textes réglementaires sont obtenus à partir de la documentation d'American Airline, disponible sur son site Web [3], en particulier celle qui décrit les termes et les conditions d'attribution des points. Ce texte constitue un document d'une dizaine de pages en anglais, destiné à un public non spécialiste puisque ce sont les clients de la compagnie. L'extrait suivant en illustre le style :

> Mileage will be credited only to the account of the AAdvantage member who flies, rents a car, stays at a hotel or earns mileage utilizing other participating companies. No mileage credit will be awarded for canceled flights or if you are accommodated on another airline.

Il s'agit de deux règles. Elles montrent bien la difficulté du vocabulaire ; Mileage will be credited est ici sensiblement synonyme de mileage credit will be awarded, mais on verra dans la section suivante (I.4) que la différence peut être importante. Participant est toujours utilisé comme synonyme de participating company, et doit être distingué de member qui désigne le client membre du programme. Dans la seconde phrase, you est encore une autre désignation du client. Enfin, les disjonctions qui énumèrent des cas différents conduisent logiquement à un ensemble de règles – une par cas. Mais dans la première phrase, la disjonction couvre tous les cas possibles et peut donc être simplifiée.

– Le cas d'usage ***Audi*** vise à certifier la conformité des procédures du constructeur automobile Audi avec la réglementation internationale sur la sécurité d'une voiture. Nous nous sommes intéressés en particulier à un texte réglementaire de l'ONU régissant les

3. http ://www.aa.com/. Nous remercions la compagnie American Airlines qui nous a permis d'utiliser ce texte, dont elle garde bien sûr le copyright

tests de qualité effectués sur les ceintures de sécurité et qui représente une trentaine de pages en anglais. C'est un texte beaucoup plus technique et difficile d'accès que celui d'AAdvantage. En voici un extrait.

> One belt or restraint system is required for the inspection of the buckle and the strength test on the buckle, the attachment mountings, the belt adjusting devices and, where necessary, the retractors. The samples to be submitted to the micro-slip test shall be kept for a minimum of 24 hours in an atmosphere having a temperature of 20 ± 5 °C and a relative humidity of 65 ± 5 per cent. The test shall be carried out at a temperature between 15 and 30 °C.

Chaque mention d'un test (inspection, strength test, micro-slip test) renvoie implicitement le lecteur à sa définition située au début du texte ; l'expression sample to be submitted to renvoie à la liste des pièces à déposer obligatoirement pour demander l'agrément. Parmi les difficultés, il faut décider s'il y a un seul strength test portant sur quatre objets, ou si ce sont quatre tests différents – le texte n'est pas constant sur le sujet – et il faut voir que le test de la dernière phrase est en fait le micro-slip test de la phrase précédente.

On voit bien pourquoi nous n'avons pas envisagé de résoudre automatiquement toutes ces difficultés et en quoi une navigation intelligente dans le texte peut contribuer à son interprétation correcte.

## I.4 Un exemple : le crédit de *miles*

Concevoir une plateforme pour aider à ce travail d'interprétation ne peut se faire qu'en relation à une méthodologie de l'interprétation, ou du moins à des éléments de méthodologie. Il s'agit alors d'organiser une coopération, sur des tâches particulières, entre l'expert et des automatisations qui ne peuvent être que partielles et imparfaites. Pour illustrer ce travail d'interprétation, nous détaillons un exemple tiré du cas d'usage AAdvantage : le vocabulaire métier y est assez familier pour que les explications soient aisément compréhensibles.

Le texte comporte 11 pages. Le passage que nous commentons est en première page ; c'est le début du 4e parmi 13 items de conditions générales pour gagner des *miles*. Cet item spécifie le montant gagné :

> AAdvantage flight mileage credit is determined on the basis of nonstop distances between the airports where your flight originates and terminates. On connecting flights, you'll receive mileage credit for each segment of your trip. On single-plane flights, you'll

receive the nonstop origin-destination mileage. [4]

### I.4.1 Le flou des termes

Une première étape concerne les variations dans l'emploi du vocabulaire et son ambiguïté. Pour interpréter le passage considéré, le "AAdvantage flight mileage credit" de la première phrase doit être identifié à "mileage credit" dans la seconde et à "mileage" dans la dernière : il s'agit du même concept dans les trois cas. Il y a d'autres variantes : on trouve dans les treize items de conditions générales (une page et demie) "AAdvantage mileage credit", "AAdvantage miles", "miles" avec un sens voisin des précédents. La difficulté de la tâche tient ici autant à la variabilité des aspects d'une même notion qu'au vocabulaire lui-même : le *mileage* représente à la fois une unité de distance (dernière phrase) et une unité monétaire (l'unité de compte du crédit). Le texte distingue aussi à l'occasion"flight mileage" (lié à un vol) et "participant mileage" (lié à un hôtel, taxi, etc.).

Au titre d'unité de compte, le *mileage* apparaît dans les mêmes items de conditions générales comme objet de différentes actions. Un outil d'extraction de relations verbales aide à inventorier ces emplois et les variations de sens qui leur sont attachées. Le crédit de miles peut "être reçu", "être gagné", "apparaître sur le bilan", "expirer" (c'est-à-dire : se périmer), "être cumulé" (*accrued*), "être proposé" (par une entreprise participant au programme), "être crédité [sur un compte]", "être accordé" (*to be awarded*), "être transférable", "être combiné entre membres", "constituer une propriété de", "être déterminé", "être accumulé" (*accumulated*), "être réclamé".

Les méthodes de construction d'ontologies à partir de textes ont montré qu'un extracteur de termes et un concordancier aident à analyser les variations de vocabulaire. L'extraction des relations verbales a aussi été utilisée par la construction d'ontologies ou la recherche d'information, souvent pour repérer des synonymies potentielles. La particularité de la situation est que les différences de sens ainsi repérées sont pertinentes dans la mesure où elles produisent des objets distincts pour les règles métier, ce qui est difficile à décider sans la coopération d'un expert métier. En ce qui concerne l'exemple, on peut considérer que le *mileage* passe par trois états :

**potentiel** comme dans les phrases citées : le *mileage* est seulement un montant qui peut être gagné ; deux passagers faisant le même voyage dans les mêmes conditions ont droit au

4. *Le crédit de* miles *des vols AAdvantage est déterminé en se fondant sur les distances sans escale entre les aéroports où votre vol commence et se termine. Sur les vols avec correspondance, vous recevrez un crédit de* miles *pour chaque segment de votre trajet. Sur les vols à avion unique, vous recevrez la distance sans escale de l'origine à la destination ;*

même crédit.

**gagné** :"On ne peut gagner du crédit pour le même vol dans deux programmes de fidélité". Le montant gagné peut et doit parfois être réclamé ; il vaut mieux garder des traces jusqu'à ce qu'il soit crédité - autrement dit, deux passagers dans les mêmes conditions gagnent chacun un crédit.

**en compte** : le crédit est porté sur le bilan du passager. Il perd son identité mais reçoit une date de péremption, qui peut par la suite être modifiée.

Cette organisation conceptuelle reflète des décisions d'architecture. Les questions de transfert entre membres, de propriété ou d'héritage sont renvoyées en dehors du SGRM considéré (sans doute à des services juridiques ou des programmes spécialisés), alors que les opérations de combinaison entre membres sont simplement impossibles. La provenance (bonus lié à un vol où à un autre service) est implicite dans des procédures en partie distinctes et n'a pas besoin d'être explicitement marquée.

A ce stade, on dispose d'une ontologie et d'un vocabulaire. Ce travail est fait dans TERMINAE, et est un préalable à la manipulation du texte dans SemEx. Nous avons ajouté un module qui projette l'ontologie sur le texte à l'aide du vocabulaire, grâce auquel la navigation peut commencer.

### I.4.2 Expliciter les règles

Dans le passage considéré, seules les deux dernières phrases décrivent des règles métier. Le critère est : tant les conditions de déclenchement que la conclusion doivent être assez précises pour que l'application de la règle soit automatisable. L'exemple illustre des indicateurs de surface assez simples. "Sur la base de" est sémantiquement flou et, même connaissant la distance entre aéroports, la première phrase ne permet aucune conclusion quant au crédit de *miles*. Les deux autres phrases sont au futur et, dans ce texte, c'est un très bon indicateur de règle - ce que l'on peut sans doute attribuer à son style juridique. Les aides à l'identification de règles sont discutées plus précisément dans la suite du mémoire, dans le chapitre V.

Nous essayons ensuite d'exprimer la règle sous la forme **Si** ...**Alors** qui est celle des SGRM. La transformation nécessite de reconnaître une partie condition et une conclusion. Une reformulation possible est :

> If you fly a connecting flight and *seg* is a segment of your trip, then you'll receive the mileage credit awarded for *seg*.

> If you fly a single-plane flight and *mil* is the nonstop origin-destination mileage then you'll receive *mil*.

Nous ne disposons pas d'une méthodologie qui contraigne une reformulation particulière ou un ordre particulier sur les reformulations successives. On remarquera que "the mileage credit for the segment" a été explicité en suivant une relation de l'ontologie. On pourrait par exemple avoir pour la première règle des étapes intermédiaires qui seraient if you fly a connecting flight, you'll receive mileage credit for each segment of your trip, puis if you fly a connecting flight, you'll receive the mileage credit awarded for each segment of your trip. La quantification (each segment) en conclusion de cette étape conviendrait à une formalisation en logique du premier ordre, mais pas pour les règles de production, pour la programmation logique ou pour les clauses de Horn avec variables qui sont les principaux formalismes employés par les SGRM. C'est pour cela que la reformulation doit être poursuivie. L'emploi de variables, qui nous semble en l'espèce facile et clair, peut aussi être évité ici si l'on veut rester au plus près du langage naturel :

> If you fly a connecting flight and a segment is part of your trip, then you'll receive the mileage credit awarded for this segment

Quelle que soit la solution prise, la transformation en **if**...**then** ne suffit pas à clarifier ces deux règles. En effet, une partie des relations sémantiques entre leurs éléments ne sont pas explicitées, parce qu'elles sont supposées connues d'un lecteur humain. Le rapport entre "connecting flight", "single-plane flight","segment" et "trip" reste implicite et les outils de la section précédente donnent peu d'indications [5]. De plus, les termes "origin" et "destination" ne sont pas reliés à "single-plane flight", et apparemment aucune des deux règles ne permet de savoir quel crédit est obtenu pour chaque segment. Il reste donc pour clarifier la formulation à expliciter ces relations entre les termes employés.

Un vol sans correspondance a néanmoins des escales. Un voyage sur ce vol peut commencer ou finir à une escale, donc il emprunte un *segment* du trajet total. Quand un voyage utilise des correspondances, il comporte plusieurs segments, chacun sur un vol différent. Le mot "vol" est ambigu : il désigne aussi bien le voyage d'un individu (ex. : conservez vos documents de vol) que l'organisation régulière d'un transport avec origine, destination, escales, horaire et identifiant (ex. : les récompenses ne sont pas disponibles sur tous les vols). On peut faire plusieurs choix d'étapes intermédiaires ; une forme raisonnablement désambiguïsée pour écrire des règles est :

5. Même si "flight" est employé 66 fois, "Connecting flight" et "single-plane flight" n'ont qu'une seule occurrence. "trip" est employé 3 fois et "segment" 14 fois. mais les deux mots n'apparaissent pas ailleurs ensemble dans le même paragraphe. Segment est utilisé parfois comme un morceau de voyage, plus souvent comme une unité de crédit de fidélité.

> If your trip is a connecting flight and *seg* is a segment of your trip, then you'll receive the mileage credit awarded for *seg*
>
> If your trip is a single-plane flight and your trip travels the segment *seg*, then you'll receive the mileage credit awarded for *seg*
>
> if *seg* is a segment and *mil* is the nonstop origin-destination mileage of *seg* then the mileage credit awarded for *seg* is *mil*

L'origine et la destination ont été rattachées au segment et la difficulté due au fait que "single-plane flight" et "segment" sont ici synonymes est levée. Ceci permet de séparer la règle qui s'applique aux vols sans escales de la procédure de calcul proprement dite. Cette dernière utilise en fait une table des distances directes entre aéroports. D'un point de vue logique, les deux premières règles peuvent être fusionnées, mais c'est une question d'implémentation et nous ne la considérons pas.

### I.4.3 Naviguer pour compléter

A ce point, le texte semble exactement traduit. Les règles métier obtenues ne sont pourtant pas encore correctes. Contrairement aux langages formels, les textes en langage naturel ne sont pas organisés pour préciser toutes les conditions en même temps. C'est ce qui se produit ici. Nous donnons rapidement les conditions supplémentaires qui portent sur le crédit de *miles* et leur position dans le texte :

- le crédit de *miles* est seulement valable pour la classe de votre billet (dans le même item) ;
- il y a un bonus de *miles* de 50% en première classe et de 25% en classe affaire (page 7) ;
- les membres qui ont le statut Elite reçoivent un crédit minimum garanti de 500 *miles* quand le parcours n'atteint pas cette longueur et est éligible (page 2 et page 7) ;
- certains tickets ne donnent pas droit au crédit de *miles* (page 7) ;
- les membres qui ont le statut Elite gagnent à ce titre pour chaque vol un bonus de *miles* qui s'ajoute au crédit de *miles* de base ou au minimum garanti (page 10)

Ajouter aux règles des conditions supplémentaires est possible, mais a l'inconvénient que les nouvelles règles sont alors justifiées par plusieurs fragments de texte éloignés. Une meilleure solution est de faire la différence pour le *mileage* potentiel entre un "*mileage* de base" défini par les premières règles et un "*mileage* ajusté" qui tient compte des bonus, du ticket, etc., et qui peut être précisé par de nouvelles règles.

Si dans cet exemple l'inventaire des règles qui interfèrent les unes avec les autres est faisable à la main, il prend du temps et est d'autant moins praticable que la taille de la collection

documentaire augmente. L'automatiser semble hors de portée. La voie qui nous a semblé la plus fructueuse est de réduire dynamiquement l'espace de recherche que l'expert explore, en lui permettant de sélectionner des fragments de texte pertinents selon des critères sémantiques riches. C'est ce qui est appelé "navigation" dans ce mémoire.

## I.5 Ce qu'apporte cette thèse

Nous proposons une plate-forme baptisée SemEx dans laquelle nous tentons de prendre en charge les principales tâches nécessaires pour guider au mieux le travail d'acquisition des règles métier ainsi que leur maintenance. Notre but est d'apporter des réponses ou des éléments de réponses pour améliorer la gestion des opérations à effectuer dans chacune des étapes d'un processus d'acquisition. Il s'agit de concevoir toute la logistique adaptée à la réalisation de ces tâches. Cette logistique concerne des outils utiles ou indispensables à la manipulation des données qu'il faut prendre en compte, mais aussi le cadre méthodologique devant guider la mise en œuvre des opérations. Sa mise en place renvoie à :

- **Une vision des tâches construite autour de l'interaction avec les acteurs** en charge de les effectuer notamment des experts métier. La gestion de chacune des tâches est centrée utilisateur, le but est d'assister celui-ci tout au long de l'accomplissement de la tâche. Par exemple, pour acquérir les règles, il est question de l'aider à comprendre facilement les textes d'origine, à les analyser et à les explorer. Pour formaliser ou maintenir les règles, il est question de mettre à la disposition de l'expert des outils qui aident à l'analyse et aux transformations.
- **Une structure de données pour organiser cette interaction**. Elle spécifie les données manipulées en entrée et en sortie ainsi que les types d'opérations supportées. Elle est définie dans le but d'assister au mieux l'acquisition et la maintenance des règles. Elle est définie autour d'une interconnexion des données à savoir les textes d'origines, les règles extraites et formalisées ainsi qu'une ontologie décrivant le vocabulaire métier utilisé. Elle est construite pour permettre dans les meilleures conditions l'exploration des textes, la fouille des règles à partir des textes, la navigation entre les différents types de données (textes et règles), la documentation et la traçabilité des données extraites, etc.
- **Une modularisation des tâches construites autour de méthodologies souples**. Elle permet de décomposer les tâches d'acquisition et de maintenance en sous-tâches. Pour cela, il s'agit d'organiser et d'encapsuler les sous-tâches interdépendantes en modules. Il y a actuellement un module de fouilles des règles, un autre de normalisation des règles

qui guident la tâche de traduction des règles identifiées en langage contrôlé, un module de maintenance. La façon dont les tâches s'enchaînent et se réalisent dans un même module est nécessairement régie par une méthodologie. Nos méthodologies permettent une organisation des opérations à effectuer dans les modules et des dépendances entre ces opérations ; elles sont définies *a minima* pour préserver la possibilité de les adapter à un domaine particulier, ou d'intégrer à la réalisation d'une tâche de nouvelles opérations et les outils de TAL associés.

– **Une implémentation partielle** de ces méthodologies oú nous avons tenté d'automatiser une partie des sous-tâches et de mettre en place un cadre permettant la réalisation manuelle simplifiée et ergonomique des autres [6]. Par exemple, nous avons rendu possible une extraction et une transformation des règles directement depuis les textes. Nous proposons aussi à l'expert métier des interfaces graphiques d'exploration et de compréhension facile des textes, qui lui permettent de valider les règles métier, et des recherches intégrées sur le texte plat ou sur la structure de données complète. Pour certaines sous-tâches actuellement non équipées, nous proposons néanmoins un inventaire des outils nécessaires pour le faire.
– **Une expérimentation** qui permet de tester nos méthodologies sur des textes réglementaires réels afin de repérer ce qui est satisfaisant et ce qu'il faut améliorer.

## I.6 Plan du rapport

La suite du mémoire est structurée en trois grandes parties :

Une première partie analyse l'état de l'art des travaux en rapport avec notre sujet et explique nos choix par rapport à l'existant. Les questions soulevées touchaient à plusieurs domaines. Nous avons organisé la diversité des sujets en deux chapitres :

– Le chapitre II est un état de l'art sur les règles métier, dans lequel nous commençons par fournir les principales définitions que nous avons trouvées dans la littérature sur les règles métier ainsi que les notions proches. Ensuite, nous étudions l'état de l'art concernant les problématiques d'acquisition et de maintenance des règles à partir de textes. Les questions sur la première problématique portent sur la façon dont les règles sont identifiées dans les textes et dont elles sont prises en compte dans un processus de formalisation. Celles sur la seconde concernent la mise à jour des règles lorsque les besoins du système d'information ou du système d'application évoluent.

6. Disponible sur `http://lipn.univ-paris13.fr/ guisse/index.php?n=Semex.Semex`

– Le chapitre III est un état de l'art sur les autres domaines sur lesquels nous nous sommes appuyés pour compléter notre analyse. Nous avons considéré les langages contrôlés qui selon la littérature jouent un rôle très important pour faciliter la formalisation des règles, et la simplification textuelle qui entre dans ce même cadre. D'autres travaux pertinents portent sur l'annotation sémantique, notamment sur l'indexation de textes par une ontologie, et sur la manipulation des graphes pour structurer les données que nous manipulons ainsi que les opérations sur celles-ci.

La seconde partie décrit et justifie notre démarche. Elle se décompose en trois chapitres :

– dans le chapitre IV, nous décrivons à grands traits l'architecture d'ensemble de notre approche et sa structuration en tâches. Nous y montrons brièvement notre vision pour gérer les tâches identifiées à travers des méthodologies et une structure de données ;
– le chapitre V entre plus en détail dans la conception des méthodologies qui organisent la gestion des tâches par la plateforme. Il y a en trois. La première concerne l'identification des règles et organise la coopération de techniques de fouille textuelle, actuellement de fouille syntaxique et sémantique. La seconde permet d'outiller le travail de formalisation des règles, et la dernière vise la maintenance dans laquelle nous comptons prendre en charge les problèmes d'incomplétude, d'incohérence et de mise à jour des règles ;
– le chapitre VI (le plus gros de cette partie) décrit les détails de la conception et du développement de la plateforme que nous proposons. Ce chapitre présente l'implémentation de SemEx avec la structure de données qu'elle utilise, les interfaces utilisateurs qui permettent de créer un cadre interactif avec l'expert métier, et les modules qui implémentent nos méthodologies et qui réunissent les outils nécessaires pour développer chacune des opérations autour de ces méthodologies.

La dernière partie décrit les expérimentations que nous avons faites et tire une conclusion de tout le travail exposé dans ce mémoire :

– Dans le chapitre VII, nous mettons en œuvre notre plateforme et nos méthodologies sur les textes réglementaires associés aux cas d'usages AAdvantage et Audi. Nous y présentons en détail les résultats de ces expériences et nous essayons d'en tirer les leçons ;
– Le chapitre VIII fait le bilan de nos contributions et dégage des perspectives d'évolution de SemEx.

# Première partie

# Prérequis

# Chapitre II

# Les règles métier

## II.1 Quelques définitions

### II.1.1 L'approche règles métier

Le terme « règles métier » [Bajec and Krisper, 2005] vient du domaine de l'intelligence artificielle où elles sont présentées comme un moyen de représenter la connaissance. Les analystes des systèmes d'information ont d'abord conçu le système en termes de structures de données et de fonctions associées à ces données. Des règles contraignent et conditionnent les opérations sur des données du système d'information. Elles sont, dans le domaine des bases de données, assimilées à des contraintes d'intégrités [Aidas and Olegas, 2009]. Au fil du temps, l'entreprise voit évoluer ses structures de données ainsi que les systèmes et les plateformes logicielles qui les utilisent. Dès lors, certaines règles disparaissent ou deviennent sans effet ou contradictoires. Cette évolution crée une nouvelle difficulté. En effet, les règles sont inclues dans les procédures système : le suivi des règles est complexe pour l'analyste qui doit trouver comment modifier la procédure ; il est difficile pour les responsables de l'entreprise qui ont la connaissance de ses règles, mais pas d'accès à la façon dont le système les applique. L'approche par règles métier [Diouf et al., 2007] est fondée sur un rapport direct entre la règle exprimée au niveau de l'entreprise et son expression dans le système par un langage de règles (voir la figure II.1 page 16). Ce rapport direct facilite la collaboration entre les gens du métier et les acteurs systèmes (concepteurs ou développeurs). Il permet de séparer la logique métier de la logique système [Halle et al., 2006]. Les gens du métier peuvent alors garder la main mise sur les règles afin de pouvoir valider ou vérifier si les règles sont en conformité avec les besoins métier de l'entreprise sans avoir à s'occuper de leur implémentation.

L'approche par règles métier est présentée par le BRG (Business Rule Group) dans un

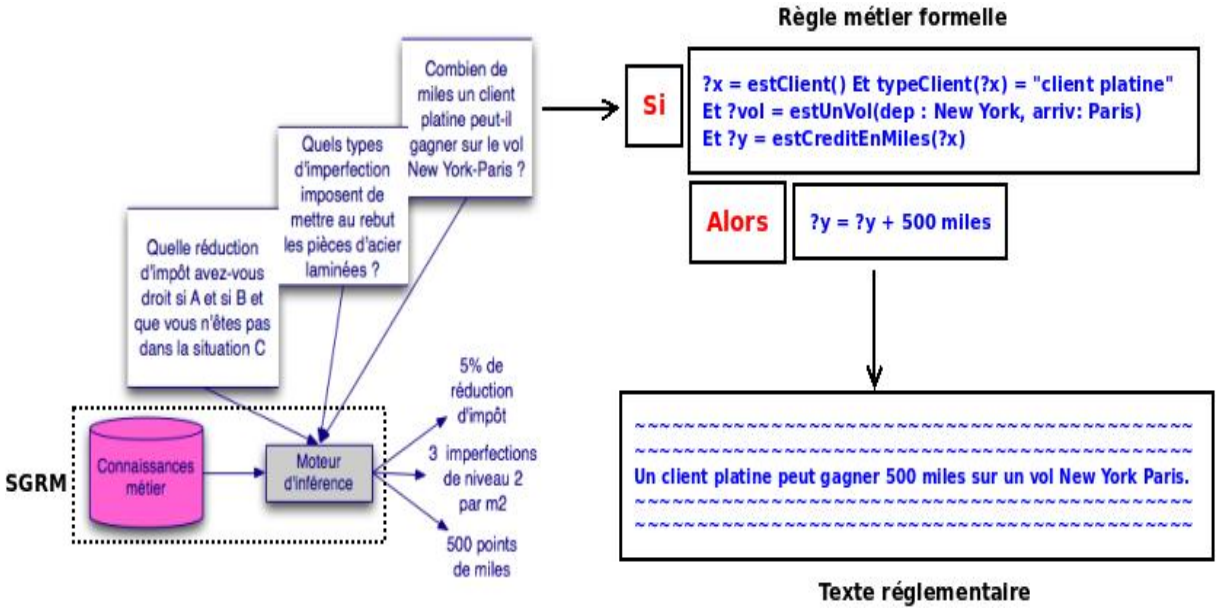



**Figure II.1** – Du texte réglementaire aux règles métier.

manifeste qui défend un certain nombre d'affirmations [BRG, 2000, Lammel, 2007] portant sur l'importance des règles métier, leur rôle et les usages auxquels elles sont associées dans le système d'information. Parmi ces affirmations on peut citer :

- Les règles sont au cœur de l'expression des besoins du système d'information et sont essentielles aux modèles métier et technologiques ;
- Les règles ne sont ni des processus ni des procédures mais elles sont intégrées dans ceux-ci. Elles servent à séparer la logique métier de la logique applicative du système d'information ;
- Les règles doivent être explicites et s'élaborer sur des faits : les concepts s'expriment par des termes ; les faits expriment des assertions sur des concepts ; les règles supportent et contraignent ces faits :
- Les règles doivent exprimer des besoins métier de l'entreprise et non des besoins techniques du système d'information ;
- Les règles doivent être exprimées clairement de façon déclarative en langage naturel à l'attention des gens du métier indépendamment de leur mise en œuvre ;
- Les règles devraient être exprimées formellement dans le système d'information de façon qu'elles restent compréhensible par les gens du métier et cohérentes par rapport aux besoins initiaux ;
- Les règles devraient être exécutables au moyen d'un moteur de règles afin de faciliter le raisonnement et la prise de décision.

### II.1.2 Qu'est-ce qu'une règle métier ?

Une règle métier [Ross, 2009a] est une règle qui n'a de sens que dans la juridiction d'une entreprise (ou d'une organisation) et qui guide la conduite et l'action de celle-ci dans la vie de tous les jours. Elle est très souvent définie sous deux angles. Considérées du point de vue des acteurs métier, les règles sont destinées à contrôler ou à influencer les comportements de l'entreprise. Et du point de vue du système d'information de l'entreprise, les règles servent à exprimer des contraintes sur la création, la mise à jour et la suppression de ses données persistantes. C'est sous ce double point de vue qu'il faut comprendre le terme « règle métier »manipulé dans le cadre du projet OntoRule (ainsi que dans cette thèse). Nous nous intéressons à la modélisation des règles métier qui sont représentées dans le système d'information de l'entreprise. Cette modélisation couvre tout le cycle de vie des règles métier dans développement du système d'information [Zachman, 1987], de leur spécification par des experts métier à leur mise en œuvre dans des applications fonctionnelles. Les règles y sont représentées sous différentes formes successives, allant d'un caractère informel à un caractère formel et automatisable. La figure II.2 , tirée des travaux du BRG (Business Rule Group [1]) [BRG, 2000], nous décrit ce cycle de vie. Elle montre que toute règle métier ("Business Rule") est définie à partir d'une formulation de règle métier ("Business rule statement") qui elle-même est définie à partir d'une politique ("Policy"). Une politique définit une directive générale de l'entreprise qui caractérise le cadre juridique associé à une règle métier (ex. : "We only rent cars in legal, roadworthy condition to our customers."). Elle peut être à l'origine d'une ou plusieurs formulations de règle métier. Chacune de ces dernières est un énoncé déclaratif d'une contrainte sur un aspect d'une politique d'une entreprise (ex. :"the car should be scheduled for service when accumulated mileage since the last service is greater than 5000, or the brakes are not satisfactory, or ..."). Une formulation de règle métier peut être à l'origine d'une ou plusieurs règles métier. Une règle métier quant à elle décrit aussi une contrainte d'un aspect d'une politique de l'entreprise. Et contrairement à une formulation de règle métier, une règle métier ne peut être décomposée en plusieurs autres règles plus détaillées. Une règle métier doit alors être atomique (ex. : "A car with accumulated mileage greater than 5000 since its last service must be scheduled for service.").

Ces trois premiers niveaux (politique, formulation de règle métier, et règle métier) constituent la partie informelle de l'usage des règles métier ; à ces niveaux les règles sont transcrites dans un langage humain compréhensible par tout expert métier. Ils précédent le niveau formel des règles métier ("Formal rule statement"), dans lequel les règles sont mises dans une forme automatisable. Pour cela, chaque règle métier est traduite en une représentation respectant une

1. http ://www.businessrulesgroup.org

grammaire formelle spécifique.

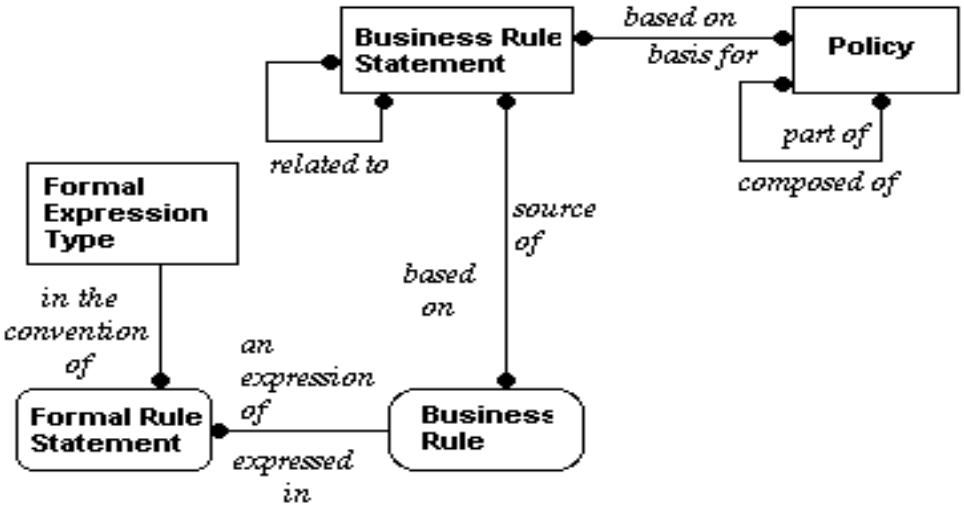



**Figure II.2** – Cycle de vie des règles métier.

Dans cette thèse, nous nous intéressons particulière au caractère atomique des règles métier (voir IV). Cette atomicité d'une règle métier peut se vérifier à travers un ensemble de cinq tests décrits dans [Ross, 2010] :

**Test 1** : une règle métier doit pouvoir être mise en pratique. Elle doit avoir un sens vital pour le système d'information de l'entreprise c'est-à-dire qu'elle s'exprime en des termes qui trouvent leur sens dans le système d'information. Le fait qu'une règle ait un sens pour l'entreprise n'implique pas forcément qu'elle soit utile pour le système d'information de l'entreprise. Il arrive qu'elle existe uniquement pour des besoins de gestion interne sans avoir un intérêt concret pour le système.

**Test 2** : une règle métier doit exprimer des faits ou des actions de l'entreprise sur les données persistantes de son système d'information. Elle ne doit pas s'appliquer sur les données propres aux systèmes qui les mettent en œuvre.

**Test 3** : une règle métier doit s'exprimer dans un langage métier et non dans un langage système. Autrement dit, un expert métier doit être en mesure de comprendre la règle sans avoir de connaissances significatives des technologies système. L'expert devrait travailler dans les termes métier et non système.

**Test 4** : une règle métier doit s'exprimer dans la juridiction de l'entreprise suivant les besoins du système d'information.

**Test 5** : une règle métier doit contrôler ou restreindre des comportements métier et faciliter la prise de décision.

Considérons les exemples suivants :

**Exemple 1** : « *Un identifiant doit être annulé si son utilisateur se loggue avec un mot de passe incorrect plus de 5 fois.*»

Cet exemple vérifie le test 1 et le test 3, il peut être mis en pratique et est facilement compréhensible par les gens du métier. Mais il ne vérifie pas le test 2, il s'exprime sur des données système, donc ce n'est pas une règle métier.

**Exemple 2** : « *Une personne qui se loggue dans le système comme un arbitre junior ne peut accéder qu'aux informations disponibles sur la base de données du serveur client.* »

Cet exemple vérifie le test 1 mais ne vérifie pas les tests 2 et 3. Il s'applique sur des données système et de plus nécessite des connaissances en langage technique système pour être compris.

**Exemple 3** : « *Un compte bancaire peut être détenu par une personne quel que soit son âge.*»

Cet exemple vérifie les trois premiers tests, il a un intérêt concret pour le système d'information, il s'applique sur des données de ce système et il s'exprime dans un langage métier compréhensible par les gens du métier. Toutefois, même s'il est conforme avec les besoins du système d'information, il ne contraint et ne restreint aucun comportement métier.

**Exemple 4** : « *An order over $1,000 must not be accepted on credit without a credit check* »

Cet exemple spécifie qu'une commande dont le montant est supérieur à 1000$ n'est acceptée à crédit que sous réserve d'une vérification de solvabilité. L'exemple vérifie l'ensemble des tests cités et donc il s'agit d'une règle métier valide.

Et enfin, on distingue trois types de règles métier atomiques [BRG, 2000, OMG, 2008] :

- **Les règles structurelles**. Un terme métier est un mot ou un groupe de mots ayant un sens spécifique pour le métier. Les données persistantes du système d'information décrivent des faits qui relient des termes métier. Une règle structurelle sert à décrire une contrainte sur un fait ("un client peut passer une commande" ou même "un client ne peut passer plus d'une commande à la fois"). Les faits correspondant à des définitions de termes ("Une voiture est un moyen de transport"), à des déclarations d'attributs (propriétés) de termes ("Une 'voiture' doit avoir une propriété 'couleur'") ou alors à des associations de termes ("location de voiture") sont aussi des règles structurelles. Logiquement, une règle structurelle est une règle qui est toujours vraie, qui décrit des propriétés toujours valides. Elle est souvent représentée sous la forme : **Si** *condition vraie* **alors** *conclusion vraie* où la condition est un fait ou une combinaison logique de faits et la conclusion est un fait.
- **Les règles opératoires** définissent une deuxième catégorie de contraintes qui s'appliquent sur les opérations de manipulation des données persistantes. Elles sont utilisées

pour mettre à jour (création, modification, suppression) les données du système d'information. Elles servent beaucoup à la prise de décision, sont déclenchées par un évènement particulier et conduisent à l'exécution d'une action. Logiquement, elles sont représentées sous la forme **Si** *condition vraie* **alors** *Action* ou **Quand** *condition vraie* **Nécessaire** *Action* (la condition et l'action constituent chacune un fait ou une combinaison logique de faits). Par exemple dans cette règle opératoire *« Une location de voiture doit être enregistrée avec un numéro de registre»*, 'location de voiture' est le fait déclencheur et l'action est l'enregistrement de cette location sous un numéro de registre.

– **Les règles dérivationnelles** sont des règles opératoires particulièrement utilisées pour les calculs ou les inférences (déduction à partir d'un fait général ou induction à partir d'un fait particulier) et qui dérivent de nouvelles connaissances (faits) à partir des connaissances existantes. Par exemple : «Le montant de l'assurance de location **est calculé à partir** du taux de l'assurance et du nombre de jours».

### II.1.3 Textes réglementaires

Un texte réglementaire est une source de représentation et de restitution de connaissances juridiques ou législatives [Jouve et al., 2001] transcrites en langue naturelle. Il constitue une matière première [Chabbat, 1997] dans le fonctionnement d'une organisation ou d'une entreprise. À nos besoins, un texte réglementaire contient des règlements qui décrivent les besoins du système d'information de l'entreprise, ils sont utilisés comme source d'information pour alimenter le système d'information. Dans ses travaux de thèse, Chabbat [Chabbat, 1997] explore, de façon détaillée les caractéristiques linguistiques et leur contexte et valeur juridique. Il est aussi possible de trouver dans [Jouve et al., 2001] des références intéressantes autour des travaux (logique déontique, ontologies, argumentation juridique, indexation et recherche d'information, représentation non formelle, etc.) sur l'élaboration de modèle de représentation et de manipulation de textes réglementaires.

Toutefois, dans cette thèse, nous appelons « texte réglementaire» tout texte qui décrit un comportement obligatoire dans l'entreprise. Au sens du droit français ça peut être une loi, un contrat, etc. Mais en gros, nous nous intéressons uniquement aux règles métier qu'ils contiennent en vue de la modélisation du système d'information de l'entreprise. Un texte réglementaire décrit alors les procédures, les instructions, les politiques, etc. qui doivent être suivis pour mettre en œuvre les besoins du système d'information. Ces textes ont la particularité selon [Jouve et al., 2001] de contenir des règlements à des niveaux variables de la juridiction de l'entreprise. Autrement dit, dans notre contexte, les textes peuvent contenir différents niveaux

de règles métier : des politiques, des déclarations de règles métier et des règles métier peuvent cohabiter dans un même texte réglementaire.

### II.1.4 SGRM

Un SGRM [2] (Système de Gestion des Règle métier) est un logiciel qui repose sur des règles métier. Il définit, exécute, contrôle et maintient les règles métier [Lammel, 2007]. Deux acteurs interagissent avec ce système pour la modélisation des règles métier : l'expert métier et l'ingénieur de la connaissance. L'expert métier est chargé de définir les règles, de veiller à leur cohérence par rapport aux besoins du système d'information de l'entreprise. L'ingénieur de la connaissance connaît le fonctionnement du système, particulièrement ses possibilités d'inférence, et dispose de son expertise pour comprendre et manipuler les règles. Un troisième acteur, l'ingénieur système ou développeur, est chargé du fonctionnement du système. Il supervise l'exécution des requêtes, surveille les indicateurs de fonctionnement (par exemple les statistiques d'usage des règles), et met à jour les règles formelles en cas d'ajout, de modification ou d'obsolescence.

Dans l'idée de conserver aux gens du métier la maîtrise des règles, une tendance récente propose de modéliser l'application à partir de règles « en langage naturel ». Le système commercial qui met ce point le plus en avant est Oracle Policy Automation. Le système SPARQLE a aussi été utilisé par IBM en recherche. De fait, le langage utilisé est très contraint ; seules sont admises les tournures qui peuvent être directement traduites en règles. Dans la mesure où l'expert sait exprimer ses connaissances dans ce langage contrôlé, les rôles d'expert métier et d'ingénieur de la connaissance sont fusionnés. Il ne s'agit toutefois pas d'utiliser les ressources en langage naturel de l'entreprise, et la source des connaissances du métier reste uniquement l'expert.

Un SGRM est un système d'aide à la décision. Les règles métier sont automatisées sous la forme : **Si** *prémisse* **Alors** *conclusion*. Cette forme générale permet des inférences différentes selon deux propriétés du formalisme : la conclusion permet seulement d'ajouter des faits, ou elle permet la modification et le retrait de certains faits ; un fait négatif dans la prémisse est satisfait quand sa version positive est prouvée fausse, ou bien quand elle est absente des données. Quelle que soit la logique sous-jacente, l'expert peut interroger le système, analyser et interpréter la réponse. Sa requête est exprimée avec la syntaxe des prémisses, la réponse est le résultat de l'application des règles.

2. http ://en.wikipedia.org/wiki/Business_rule_management_system

Lämmel [Lammel, 2007] a fait un état de l'art sur les composants d'un SGRM et sur les différents SGRM existants. Un SGRM fournit, en général, un ensemble de modules pour la définition, la manipulation et la maintenance des règles métier, soient :

- un module d'acquisition composé d'un éditeur de règles. Celui-ci offre parfois la possibilité aux experts de définir les règles en langage naturel ou à l'aide d'une interface de menus déroulants avant qu'elles soient implémentées par les développeurs ;
- un module de stockage des règles implémentées met en œuvre une base de règles où sont stockées les règles opératoires. Les règles structurelles et les faits sont quant à eux stockés dans les bases de données du système d'information de l'entreprise ;
- un module d'exécution et de déploiement des règles ;
- un module d'interrogation, d'interprétation et d'analyse des résultats qui aide l'expert pour les prises de décision ;
- un module de maintenance à partir duquel les mises à jour de la base de règles sont effectuées lorsque le système d'information évolue ou lorsque des incohérences et inconsistances apparaissent dans la base de règles.

Lämmel propose une étude comparative sur les SGRM les plus en vue, à travers un ensemble de critères d'évaluation portant sur l'utilisabilité et les performances des différents modules d'un SGRM. Il évalue les systèmes Blaze Advisor [3], ILOG JRules [4], JBossRules [5], Oracle-Rules [6], mandarax [7], QuickRules [8] et Visual Rules [9]

Dans le cadre du projet OntoRule, le SGRM JRules a été utilisé avec la participation du partenaire IBM France. Des travaux de mise en œuvre de règles métier dans JRules ont été effectués notamment concernant le système d'inférence sous-jacent. Plus proches des questions d'acquisition, les travaux de Amina Chniti [Chniti et al., 2012] traitent de l'impact de l'évolution des ontologies OWL sur les règles formulées à l'aide de ces ontologies.

3. www.fico.com/fr/Solutions
4. www-01.ibm.com/software
5. labs.jboss.com/portal/jbossrules
6. www.oracle.com
7. www.mandarax.org
8. www.yasutech.com
9. www.visual-rules.de

# II.2 Acquisition et maintenance des règles métier

## II.2.1 Plates-formes d'acquisition

A notre connaissance , il n'existe pas de plateformes spécifiques à l'acquisition des règles en vue de mettre ensuite en œuvre des SGRM. Le développement des règles métier s'intègre plutôt au cycle du développement de ces logiciels. Elles ne sont définies que pour une application bien donnée. L'acquisition des règles est gérée à plusieurs niveaux du cycle de vie de développement de l'application et par plusieurs acteurs ayant des expertises différentes. En effet, l'architecture du cycle de vie repose sur une séparation entre la logique métier et la logique système de façon à permettre une meilleure collaboration entre experts métier et ingénieurs [Halle et al., 2006]. L'architecture la plus utilisée est celle proposée par l'OMG (Object Management Group) et appelée MDA (Model Driven Architecture) [Miller and Mukerji, 2003]. Le MDA [Martínez-Fernández et al., 2008] est une démarche orientée modèles qui permet de gérer tout le cycle de vie d'une application, de sa spécification à son développement. Le MDA vise une interaction entre les acteurs du développement mais aussi une interopérabilité des différentes phases du cycle ainsi que leur portabilité et leur utilisabilité.

Le MDA [Miller and Mukerji, 2003] est constitué de trois modèles (voir Figure II.3) qui vont servir à modéliser l'application et par transformations successives à générer le code source de celle-ci :

- le CIM (Computation Independent Model) permet de modéliser le système d'information indépendamment de l'application informatique. Ce modèle décrit tout le vocabulaire métier de domaine et les règles métier nécessaires à la spécification des besoins et contraintes du système d'information. Ce modèle est défini par les experts ou analystes métier qui, usant de leur expertise du domaine, décrivent le cahier de charges de l'application visée sans s'occuper des détails de son implémentation. Les ressources définies à ce niveau, comme elles sont indépendantes, peuvent être partagées par plusieurs applications du système d'information. Ces ressources ont la particularité d'être transcrites en langage humain et très souvent sous forme textuelle ;
- le PIM (Platform Independent Model) intègre les phases d'analyse et de conception de l'application et correspond à sa logique métier. Il est aussi indépendant de l'application et de ses détails techniques mais c'est à ce niveau que le modèle PIM est utilisé pour représenter formellement le modèle CIM. Ce travail est souvent effectué par des ingénieurs spécialistes en conception d'applications informatiques, et il consiste concrètement à traduire les ressources en langue naturelle produites par le CIM dans un langage formel ;

– le PSM (Platform Specific Model) est le seul modèle qui dépend de l'application et est spécifique à celle-ci. Il sert à implémenter le modèle PIM en langage machine. C'est à ce niveau que des ingénieurs développeurs se procurent le modèle PIM pour l'adapter à l'application suivant les informations techniques de celle-ci avant de générer le code source associé.

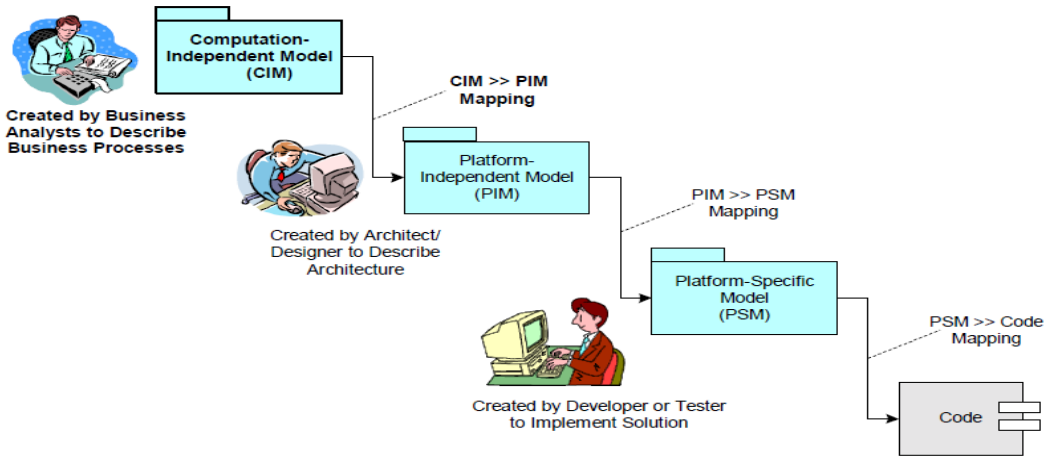



**Figure II.3** – Schéma OMG du modèle MDA.

L'objectif de l'architecture MDA est de fournir tous les outils nécessaires à la construction du modèle CIM et de ses transformations pour produire les modèles PIM et PSM :

1. **CIM** : des outils d'édition sont utilisés pour permettre aux experts métier de spécifier le modèle en langue naturelle ;
2. Passage **CIM à CIM** : des outils de raffinement de la langue naturelle du CIM dans le but de la simplifier, de l'enrichir, de la préciser ;
3. Passage **CIM à PIM** : des outils de transformation de la langue naturelle en langage formel de représentation du modèle CIM ;
4. Passage **PIM à PIM** : des outils de raffinement le PIM en vue de le filtrer et le spécialiser afin de s'approcher au mieux de l'application visée ;
5. Passage **PIM à PSM** : des outils pour ajouter au modèle PIM les informations techniques spécifiques à l'application ;
6. Passage **PSM à PSM** : des outils de réalisation du PSM en code source machine, et de déploiement de tous les composants systèmes.

### II.2.2 Acquisition à partir de textes

Dans cette thèse, nous nous intéressons essentiellement aux trois premiers types d'outils proposés par le MDA (notamment pour le CIM et les passages CIM à CIM et CIM à PIM) sur l'acquisition des règles métier. Cela a fait l'objet de plusieurs travaux de recherche [Bajec and Krisper, 2005, Halle et al., 2006] qui se sont intéressés en particulier au cycle de vie des règles entre le CIM et le PIM. Ce cycle correspond à l'évolution des règles d'un état initial où elles sont incluses dans les documents de spécification (textes réglementaires) du système d'information, à un état final où elles sont automatisées et mise en œuvre dans le SGRM. Les travaux s'interrogent alors sur un certain nombre de points à savoir 1) l'acquisition des règles à partir des textes réglementaires, 2) leur modélisation dans un langage formel afin de faciliter leur automatisation, 3) leur intégration dans le SGRM (stockage, exploitation, maintenance).

Nous nous concentrons essentiellement sur l'acquisition et la modélisation des règles. Autrement dit sur la question de la prise en compte des documents pour acquérir et modéliser les règles qu'ils contiennent. Cette question, déjà soulevée dans [BRG, 2000, Halle et al., 2006, Dubauskaite and Vasilecas, 2009], pose un problème de transformation de connaissances de l'informel au formel c'est-à-dire de traduction des règles contenues dans des documents rédigés en langage humain en des règles formalisées. Les recherches montrent que ce passage ne peut être effectué de façon automatique, du fait de l'écart entre la complexité de la langue et l'expressivité fournie par les langages formels.

[Dubauskaite and Vasilecas, 2009] inventorie différents éléments qui peuvent intervenir dans le processus d'élicitation tel qu'il est pratiqué : 1) l'identification des règles à partir de dialogues ou de documents et l'écriture de spécifications en langage naturel 2) l'utilisation de langages naturels contrôlés et de patrons de règles pour écrire des spécifications 3) la modélisation graphique des spécifications, par exemple en UML ou OCL 4) leur implantation en code exécutable. On retrouve les trois niveau du MDA, mais 1) et 2) constituent une alternative et non des processus parallèles.

#### II.2.2.1 Identification des règles métier

L'identification consiste à sélectionner les fragments de texte qui expriment les règles dans les textes réglementaires.

Les fragments sélectionnés correspondent très souvent à des phrases. Pour marquer qu'elles ne sont pas encore validées, elles sont appelées dans [Tobon and Franco, 2010], « phrases candidates de règles métier » ('business rule candidates' sentences). L'appellation devient « règles candidates » (RC) une fois qu'elles sont validées règles métier. Leur identification est une

tâche classiquement allouée aux analystes ou aux experts de règles métier. Ils les filtrent rapidement à la main mais cette tâche devient très vite complexe du fait des gros volumes de textes à consulter. Pour assister les experts, des outils de détection automatique sont proposés [Tobon and Franco, 2010]. Ce sont des outils de traitement automatique de la langue (TAL) qui reposent sur des méthodes d'extraction automatique basées sur des analyses syntaxiques et sémantiques de textes. Ces outils tentent de détecter les phrases candidates suivant des caractéristiques linguistiques. [Tobon and Franco, 2010] ont distingué deux approches pour la détection automatique :

**l'approche Statistique** est très peu utilisée. Elle repose sur des calculs statistiques basés sur la fréquence des mots dans le texte. Il s'agit alors de considérer les mots les plus fréquents jusqu'à un certain seuil et ceux-là sont appelés clés candidates. Des outils TAL sont utilisés pour l'analyse syntaxique et sémantique afin de structurer les textes et supprimer les ambiguïtés et affinités lexicales dans la liste des clés candidates. Ensuite toutes les phrases contenant les clés candidates sont considérées comme phrases candidates et devront être validées par l'analyste métier ;

**l'approche linguistique** est l'approche la plus utilisée. Elle consiste à repérer les phrases candidates à partir d'un ensemble de patrons linguistiques de règles métier. Un patron est une règle d'extraction ou une expression régulière qui est appliquée sur un texte et qui en extrait tous les fragments textuels vérifiant le patron. Chaque patron (Tableau II.1) est défini à partir des mots clés caractéristiques ou descriptifs de règles métier qui sont proposés par exemple dans SBVR [OMG, 2008], RuleSpeak [Ross, 2009b], mais aussi à partir d'heuristiques grammaticales (HG) portant sur les structures de la langue. Les patrons sont alors appliqués sur les arbres syntaxiques [10] des phrases du texte, obtenus après l'analyse syntaxique du texte. Le résultat est un ensemble de phrases candidates.

Tobon et al. [Tobon and Franco, 2010] proposent un outil de détection automatique de règles métier à partir de textes réglementaires rédigés en langage naturel. Le BRElicitationTool utilise l'outil de TAL Stanford Parser pour effectuer l'analyse linguistique des textes. Il utilise aussi l'outil Tregex [11] pour appliquer les patrons linguistiques sur l'ensemble des arbres syntaxiques(chacun correspondant à une phrase des textes) résultant de l'analyse syntaxique. Les patrons sont définis sur la base de la structure d'une phrase, de la catégorie syntaxique de chaque mot de la phrase et sur un ensemble de mots clés caractéristiques de règles métier. Les mots-clés sont proposés dans SBVR [OMG, 2008], RuleSpeak

10. http ://fr.wikipedia.org/wiki/Arbre_syntaxique
11. The Stanford Natural Language Processing Group

| | |
|---|---|
| SBVR | **each...**<br>**...at least one...**<br>**if p then q**<br>**...must...** |
| RuleSpeak | **may...only if**<br>**must be computed as**<br>**must be considered...if** |
| HG | **When + phrase + then/implies + phrase**<br>**Subject + will + verbal phrase**<br>**If + phrase + then/implies + phrase** |

**Tableau II.1** – Patrons linguistiques basés sur des mots-clés et des heuristiques grammaticales.

[Ross, 2009b]. Tregex applique alors un patron du genre "NP...can...only...VB...if...NP" et extrait toute les phrases comme celle dont l'arbre syntaxique est représenté dans la figure.

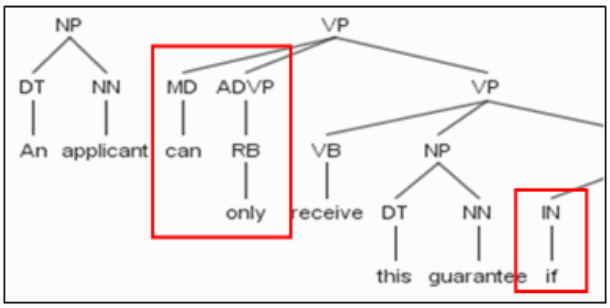



**Figure II.4** – Un exemple d'arbre syntaxique de phrase relatant une règle métier.

BRElicitationTool [Tobon and Franco, 2010] a été appliqué sur un exemple de texte réglementaire qui a aussi été analysé par des étudiants et professeurs, qui ne sont pas forcément des analystes métier, et qui ont détecté manuellement les règles métier candidates sur la base des mots clés SBVR et RuleSpeak [12]. Dans cette expérience, les règles candidates détectées manuellement avaient un rappel de 28,42% et une précision 68,44%. Ce résultat est de loin moins bon que celui de l'outil qui fournit un rappel de 72,97% et 71,05% de précision. Ces résultats ne sont pas forcément interprétables sur tous les types de textes règlementaires mais ils montrent la valeur ajoutée de la détection automatique.

Toutefois, suivant la complexité et le caractère ambigu de la langue cette méthode nécessite une intervention humaine (souvent celle de l'analyste) pour valider si une règle

12. Le fragment de texte est un extrait des documents réglementaires du projet "The SHARING Project" qui vise à développer un système d'information traitant des prêts hypothécaires d'une organisation, ING, basée aux Pays-Bas.

candidate extraite est une règle métier ou pas. L'identification aboutit à un ensemble de règles candidates valides.

#### II.2.2.2 Formalisation des règles métier

La formalisation a fait l'objet de plusieurs travaux [Lovrencic et al., 2006, Braye et al., 2006]. Elle est une étape intermédiaire dans l'automatisation des règles afin de faciliter leur passage du langage humain à leur forme implémentée.

Concrètement, la formalisation des règles [Aidas and Olegas, 2009] consiste en leur traduction dans des langages formels (LF) ou dans des modèles conceptuels. Parmi ces derniers on peut citer les modèles entités-associations (BIER (Behaviour Integrated Entity-Relationship) [Eder et al., 1986]), les modèles orientés objets [Lukichev and Jarrar, 2009] (UML [13](Unified Modeling Language), OCL [14] (Object Constraint Language), ORM [15] (Object Role Modeling)) et les langages formels définis pour le Web et le Web sémantique (OWL, SWRL, RuleML, RIF, F-logic, etc) [Cisternino et al., 2009].

Toutefois, lorsque les règles existent initialement en langue naturelle, leur traduction formelle ne peut se réaliser de façon directe du fait de l'écart. Nous nous sommes intéressé à cette problématique du passage du langage naturel au langage formel dans la littérature. Selon les travaux de Bajec et Krisper dans [Bajec and Krisper, 2005], la traduction formelle des règles fait intervenir deux types d'acteurs, l'expert métier et l'ingénieur système. L'expert est du domaine et il est chargé de spécifier les besoins de l'entreprise sous formes de règles métier. L'expert rédige ces règles sous forme textuelle en langue naturelle. Et ensuite, les règles sont formalisées avant d'être implémentées en code machine par l'ingénieur afin de permettre leur automatisation dans les SGRM. L'analyse de Bajec et Krisper fusionne donc les rôles d'ingénieur de la connaissance et d'ingénieur système.

Ces travaux ont aussi montré que le passage des règles du langage naturel au langage formel est complexe et qu'il n'est pas automatique, ce qui a des conséquences dans le suivi et la maintenance des règles :

– le manque d'expertise du domaine métier et la complexité de la langue naturelle rendent l'écriture formelle des règles très difficile pour l'ingénieur. La manipulation des textes nécessite une certaine connaissance du domaine ; leur variabilité syntaxique et le caractère informel de leur sémantique complique la formalisation et encore plus l'automatisation de celle-ci ;

13. http ://www.uml.org
14. http ://www.omg.org/cgi-bin/doc ?formal/2006-05-01/
15. http ://fr.wikipedia.org/wiki/Mapping_objet-relationnel

- les experts qui rédigent ces règles sont très fidèles aux besoins de l'entreprise et ne prennent souvent pas en compte leur utilisation par le SGRM. Cela pose problème à l'ingénieur qui peut difficilement concilier les règles fournies par l'expert et la mise en œuvre des règles au niveau système ;
- une fois formalisées, les règles sont écrites dans un langage formel de règles métier. Dès lors, elles deviennent incompréhensibles par l'expert qui éprouve à son tour des difficultés pour les valider, les expliquer, vérifier leur conformité avec ses règles ou pour les modifier quand les besoins de l'entreprise évoluent.

Les langages contrôles ont été proposés [BRG, 2000, Halle et al., 2006], [Dubauskaite and Vasilecas, 2009] comme une solution à mi-chemin entre langage naturel et langage formel, surtout pour améliorer la collaboration entre les experts et les ingénieurs. Un langage contrôlé [Schwitter, 2005] restreint la grammaire et le vocabulaire utilisés par la langue naturelle afin de réduire son caractère ambigu et complexe. Il est surtout connu pour améliorer la lisibilité du langage naturel en fournissant une version simplifiée et facilement compréhensible de celui-ci. Le langage contrôle offre des structures linguistiques qui permettent d'écrire les règles sous une forme syntaxiquement structurée, précise au niveau lexical et non ambiguë dans son ensemble.

### II.2.3 Maintenance des règles métier

De la même manière que l'acquisition des règles métier est un processus inclus dans le cycle de développement d'une application, la maintenance des règles en fait partie. La maintenance est prise en compte dans l'architecture MDA [Miller and Mukerji, 2003] pour assurer l'évolution de l'application et fait aussi l'objet de plusieurs travaux comme [Wan-Kadir and Loucopoulos, 2004]. L'objectif de cette architecture reste cependant de fournir les outils nécessaires à implémenter les règles pour leur prise en charge par les ingénieurs : dans le MDA, le processus de maintenance n'intervient qu'une fois les règles implémentées et automatisées. De ce fait, la maintenance n'est lancée qu'au niveau du modèle PIM et sous la responsabilité des seuls ingénieurs. En conséquence, par manque d'expertise métier, seuls les problèmes de maintenance qui sont détectés au niveau de l'application sont pris en compte [Paige et al., 2005]. Les autres problèmes, indépendant de l'application, décrivent des problèmes au niveau du modèle CIM ou du passage entre le CIM et le PIM. Ces problèmes (voir section suivante), qui nous intéressent ici, interpellent les experts métier.

Le module de maintenance qui est inclus dans le MDA est le plus souvent constitués de trois outils pour :

- la détection : Les ingénieurs doivent disposer d'outils pour détecter les problèmes devant mener à la maintenance des règles ;
- la verbalisation [Halpin, 2004] : les règles formelles sont retraduites dans un langage contrôlé ([Wagner et al., 2005]) manipulable par les experts. Pour cela, tous les langages contrôlés (voir III.1) utilisés pour la représentation des règles peuvent servir à verbaliser leur version implémentée ;
- la mise à jour : les experts doivent disposer d'outils pour corriger les problèmes détectés en modifiant les règles affectées. Les règles repassent ensuite dans le processus de formalisation avant d'être implémentées, et l'application sera ainsi maintenue.

Concrètement, la gestion de la maintenance repose sur des outils de détection des problèmes et de mise à jour des règles suite à la correction de ces problèmes. Des travaux ont été effectués dans le cadre du projet OntoRule [Fink et al., 2011] et des propositions de méthodes ont été faites. Ces méthodes montrent que la gestion de la maintenance pose trois questions auxquelles l'expert devra apporter successivement des éléments de réponse pour assurer cette tâche.

1. Problèmes. **Quels sont les problèmes de maintenance qui sont détectés dans les règles ?** à ce niveau, il s'agit d'analyser syntaxiquement ou sémantiquement les règles afin d'identifier celles qui nécessitent une modification. Les problèmes les plus fréquents se situent soit au niveau de la terminologie métier utilisée dans l'écriture des règles (voir [Chniti et al., 2012]) soit au niveau de la base de règles. Les problèmes dans la base de règles sont en général de deux types : les problèmes d'incohérence ou de contradictions entre les différentes règles, et les problèmes d'évolution notamment quand les besoins de l'entreprise changent ou que des règles deviennent obsolètes.

2. Diagnostic. a) **Comment capturer l'origine d'un problème donné ?** et b) **Qu'est-ce qui explique ou justifie ce problème ?** à ce niveau, le but est de découvrir l'origine des problèmes dans le cycle de développement de l'application pour ainsi dégager les responsabilités. En effet, à partir des règles traduites en langage contrôlé (verbalisées), les experts cherchent à savoir si le problème provient de la description des règles à leur niveau dans le cycle (modèle CIM). Pour ce faire, ils vérifient la conformité des règles affectées par le problème avec les besoins initiaux de l'entreprise. Cette vérification permet d'isoler les règles incorrectes.

3. Action de résolution. **Quelles opérations de mise à jour doivent être effectuées sur les règles ?** à ce dernier niveau, il s'agit de corriger tous les problèmes et de mettre ainsi à jour les règles incriminées lors du diagnostic. Une fois les corrections effectuées par les experts sur les règles en langage contrôlé, celles-ci devront repasser dans le processus

de formalisation (modèles PIM et PSM).

## II.3 Règles métier et exigences

Les règles métier sont aussi beaucoup utilisées dans le domaine de l'ingénierie des exigences. Ces dernières sont (dans la formulation de wikipedia [16]) l'expression d'un besoin documenté sur ce qu'un produit, un logiciel ou un service informatique, devrait être ou faire. Elles sont très souvent assimilées au cahier des charges dont la production est un prérequis pour les étapes de conception et de développement d'un logiciel.

L'acquisition et la maintenance d'exigences a aussi fait l'objet de recherches [Lapouchnian, 2005], [Van Lamsweerde, 2001] . Elle a pour objet de gérer, d'établir et de maintenir un accord avec les parties prenantes sur les exigences de l'application à développer. Ces activités se regroupent autour de la spécification des exigences et de la maintenance de la cohérence entre elles lors qu'elles subissent des changements. La spécification des exigences consiste de façon itérative à comprendre le problème et élucider les besoins, à transformer ces besoins en exigences, à documenter ces dernières et à valider leur conformité avec l'application attendue.

Pour l'ingénierie des exigences, les règles métier sont considérées comme des «citoyens de premières classes» [17]), elles font partie des exigences fonctionnelles métier (qui précédent les exigences techniques) qui décrivent les caractéristiques des fonctionnalités que le produit doit exécuter. Cependant, toutes les exigences fonctionnelles ne sont pas des règles métier. Les exigences peuvent laisser un degré d'indétermination quant aux décisions qui peuvent être prises, alors que les règles mettent en œuvre des décisions opérationnelles dans les SGRM. Par exemple une exigence peut servir à décrire une politique de l'entreprise, qui se borne à réduire l'éventail des décisions possibles, ce qui ne peut se faire avec une règle métier qui doit déterminer une décision.

Deux autres différences entre règles métier et exigences méritent d'être signalées. D'abord les exigences sont l'objet d'une négociation entre les parties prenantes pour spécifier le logiciel qui doit être implémenté. Les règles métier pour leur part sont créées par l"entreprise seule, sans l'avis des autres acteurs. Dans notre contexte, la conformité totale des règles métier aux documents de l'entreprise doit être assurée, alors que pour les exigences, la négociation entre acteurs peut amener à laisser de coté certains aspects. En second, les exigences une fois recueillies servent à tester la conformité du logiciel, alors que les règles métier sont implémentées comme

16. http ://fr.wikipedia.org/wiki/Exigence_(ingénierie)
17. "Rules are a first-class citizen of the requirements world" Leçon 5 du cours sur les règles métiers à l'ICIS, Université de Nijmegen. https ://lab.cs.ru.nl/BusinessRules/Requirements_engineering

parties intégrantes de ce logiciel. Les règles seront ainsi utilisées pour prendre des décisions sur de nombreux cas particuliers, alors que les exigences ne serviront à tester qu'un seul logiciel.

Au titre des similitudes, les exigences comme les règles métier sont prises en charge notamment à partir de documents d'exigences écrits en langue naturelle. Et on retrouve une séparation des acteurs analogue : pour [Van Lamsweerde, 2001] la modélisation des exigences est faite par des experts de domaine avant que ces derniers passent la main aux ingénieurs chargés du développement de l'application cible. C'est pour cette raison que l'ingénierie des exigences vise un contexte social favorisant la communication entre les différentes parties prenantes dans la spécification des exigences [Nuseibeh and Easterbrook, 2000]. Son objectif est d'avoir un vocabulaire commun et un langage commun entre les parties. Toutefois, l'une des principales difficultés [Lapouchnian, 2005] est l'imprécision, la nature informelle des exigences et le fait qu'elles soient produites à partir d'observations informelles du monde réel pour être traduites dans des langages de spécification formels.

C'est ainsi que des méthodologies de spécification [Van Lamsweerde, 2001], [Nuseibeh and Easterbrook, 2000, Jackson, 1997] ont été proposées pour amener les différentes parties à produire interactivement les exigences de façon compréhensible par tous. Les principales méthodologies sont KAOS (Knowledge Acquisition in Automated Specification) [Shaw, 1990] et CREWS (Cooperative Requirements Engineering With Scenarios) [Ralyté, 1999]. Elles reposent sur ma construction d'un modèle des exigences comportant notamment les définitions des concepts manipulés et un modèle graphique pour l'organisation des buts et des besoins. Ces méthodologies fournissent un document structuré d'exigences à partir duquel peut commencer le travail de conception et de traduction formelle par les ingénieurs. Ce document présente en principe des exigences cohérentes et non ambiguës et contient toutes les exigences nécessaires à la conception de l'application cible. L'utilisation d'un modèle graphique est analogue à l'utilisation d'UML ou OCL pour les règles, mais la démarche est descendante (du but principal vers les exigences détaillées) alors que pour les règles elle est ascendante.

Dans ces méthodologies, le passage des documents de l'entreprise aux exigences est pris en charge par les experts en dehors de la plateforme. Nous n'avons pas trouvé de travaux qui ont expérimenté l'utilisation des méthodologies de spécification à partir des documents de l'entreprise. Les seuls travaux à notre connaissance qui se sont orientés dans ce sens portent sur les relations entre l'ingénierie des exigences et des systèmes de traitement automatique de la langue (TAL) [Huyck and Abbas, 2000, Berzins et al., 2008, Barcellini et al., 2012]. Pour ces travaux la question n'est pas d'identifier les exigences dans les documents de l'entreprise mais d'utiliser des techniques TAL pour détecter les ambiguïtés syntaxiques dans leur reformulation afin d'éviter que des erreurs soient commises à la conception. Les mêmes techniques sont utilisables

pour vérifier la formulation des règles métier dans un langage contrôlé.

# Chapitre III

# Autres notions utiles

## III.1 Les langages contrôlés

### III.1.1 Présentation générale

Les langages contrôlés [1] sont des sous-ensembles des langues naturelles dont les vocabulaires et grammaires ont été limités afin de réduire les ambiguïtés et les difficultés linguistiques. Certains sont conçus pour jouer un rôle de médiation entre les humains, d'autres entre les humains et les machines [Schwitter, 2010]. Dans le premier cas, ils servent à améliorer la lisibilité et l'intelligibilité des documents techniques pour des lecteurs qui peuvent ne pas être techniciens du domaine. Et dans le second cas, les langages contrôlés améliorent le traitement informatique d'un texte notamment pour l'acquisition et la représentation des connaissances qu'il contient. Notre travail se situe dans ce dernier cas où l'utilisation d'un langage contrôlé permet d'avoir un langage intermédiaire à la fois compréhensible par les experts métier qui rédigent les règles et par les ingénieurs qui les automatisent. R. Schwitter [Schwitter, 2010] cite différents domaines où les langages contrôlés sont utilisés dans cette optique de collaboration « humain-machine » : le Web sémantique avec la construction d'ontologies formelles, l'ingénierie des exigences avec la rédaction des cahiers de charges, mais aussi les règles métier. Ces langages contrôlés servent à faciliter la traduction du langage naturel vers un langage formel. Ils ont l'avantage de pouvoir se traduire automatiquement dans des représentations logiques comme la logique du premier ordre et les logiques de description, qui définissent la base de plusieurs langages formels de règles métier.

L'utilisation des langages contrôlés, qui nous intéresse ici, découle du problème lié à la formalisation des règles métier comme présenté dans II.2.2.2. Dans ce contexte, plusieurs lan-

1. https ://sites.google.com/site/controllednaturallanguage/

gages contrôlés ont été proposés (voir [Wagner et al., 2005]). La plupart d'entre eux (comme ACE, Metalog, I1CE-V1) sont essentiellement utilisés pour la verbalisation de modèles formels (comme UML, OCL, etc.). Ils sont utilisés pour aider à la mise à jour des systèmes d'information : ils permettent de traduire en langage naturel les règles métier que décrivent ces modèles, afin qu'elles puissent être modifiées et validées par les experts. SBVR se veut d'abord un modèle de représentation des règles indépendant du langage. Il est moins formel que UML ou OCL, puisqu'il reste une représentation textuelle. Le modèle de représentation SBVR dispose de plusieurs notations : le langage contrôlé SBVR-SE (SE pour *Structured English*), des notations graphiques comme RuleSpeak ou ORM.

### III.1.2 SBVR

SBVR [OMG, 2008] est un standard du groupe OMG. Ce n'est pas un langage mais c'est un méta modèle : il standardise la description déclarative sous forme de faits et de règles du fonctionnement d'une entité complexe comme une entreprise. Il sert à définir le vocabulaire et la grammaire nécessaires à la documentation sémantique des connaissances métier de l'entreprise. Il est conçu dans une optique de formalisation de règles, autrement dit en pensant que la représentation formelle est le but et non la source de la représentation en SBVR[Spreeuwenberg and Healy, 2009]. Selon ses concepteurs :

- SBVR est le fruit de plusieurs travaux de recherche qui se sont aussi intéressés aux questions liées à la modélisation de connaissances métier et à la collaboration entre expert et ingénieur.
- Il permet d'assister les experts dans la spécification et la définition du modèle sémantique de domaine (tels que le vocabulaire et les règles métier).
- Il permet de représenter un modèle de domaine en langue naturelle de façon suffisamment claire, précise et désambiguïsée afin qu'il soit facile à formaliser.
- Il spécifie diverses structures linguistiques afin de couvrir un plus grand nombre de langues.
- Il permet de structurer [2] toutes les connaissances métier de l'entreprise. Les vocabulaires métier sont définis avec différents niveaux d'expressivité (terminologique, taxonomique, thésaurus, etc.). Les règles métier quant à elles s'appliquent sur ces vocabulaires, les règles structurelles définissant les contraintes sur les concepts et les règles opératoires et dérivationnelles définissant les opérations de manipulation et de mise à jour.

2. http ://www.omg.org/news/meetings/tc/mn/special-events/br/

– Il permet de rendre les connaissances métier accessibles à la fois pour les experts et pour les ingénieurs ; il permet aussi aux entreprises et systèmes logiciels de communiquer et partager leurs connaissances métier.

Nous ne nous occupons pas ici de la définition du vocabulaire métier, nous nous focalisons uniquement sur l'usage de SBVR-SE comme langage intermédiaire pour l'écriture des règles métier. La norme indique des méthodes d'interprétation des formulations SBVR en logique du premier ordre étendue par une modalité déontique (Autorisé / Obligatoire). Cette sémantique utilise un vocabulaire métier défini à part des règles, et qui constitue l'univers d'interprétation. Elle est neutre quant à l'emploi en totalité ou en partie de la négation par l'échec, qui dépend de l'application. Le vocabulaire (voir Figure III.1) regroupe l'ensemble des termes métier ou nom de concepts, des entités nommées, des verbes qui traduisent des faits liant les concepts, et un ensemble de symboles linguistiques (de l'anglais) :

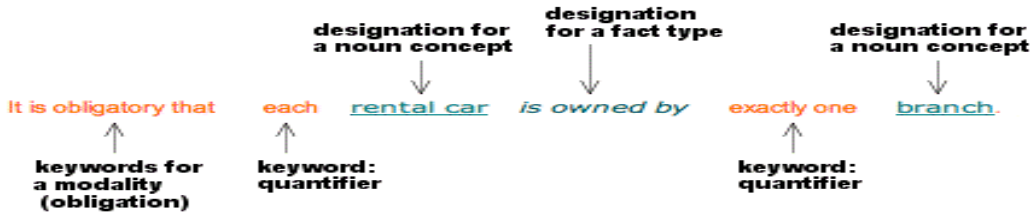



**Figure III.1** – Ecriture d'une règle métier sous SBVR-SE.

**concept**

Le style 'concept' (vert souligné) est utilisé pour désigner un terme qui est un nom de concept correspondant à une entité typée du vocabulaire métier (ex : 'person', 'rental car').

**Nom**

Le style 'Nom' (vert double souligné) est utilisé pour désigner des instances de concept comme des entités nommées correspondant à des noms propres de personnes ('Obama') ou de lieux ('California'), et aussi à des chiffres ('25 years').

***verbe***

Le style 'verbe' (bleu italique) est utilisé pour désigner les faits, souvent représenté par des verbes, entre des noms de concept ou des entités nommées.

**mot-clé**

Le style 'mot-clé' (rouge) est utilisé pour désigner les symboles linguistiques anglais utilisés pour construire une phrase. Il s'agit de tous les mots de la langue autorisés autres que ceux du vocabulaire métier. Un bon nombre d'entre eux apparaissent dans les formulations

logiques et les autres sont indispensables à la construction de la règle (ex :'**the**', '**that**', '**another**', '**a given**', '**who**', '**of**', '**what**', etc)

SBVR propose plusieurs types de formules logiques, dont les plus utilisés sont :

- Les formulations construites avec des opérateurs logiques pour exprimer une structure de négation (it is **not** the case that $p$), de conjonction ($p$ **and** $q$), de disjonction ($p$ **or** $q$), d'implication (**if** $p$ **then** $q$, $q$ **if** $p$), d'équivalence ($p$ **if and only if** $q$), etc.
- Les formulations construites avec les opérateurs modaux modifient la modalité d'un fait en le rendant obligatoire, possible ou toujours possible. Elle utilise des expressions qui sont insérées en en-tête de règle ("**It is obligatory that**" dans une règle opératoire, "**It is necessary that**" dans une règle structurelle).
- Les formulations de quantification permettent de quantifier un concept : **a, an, each** (quantification universelle), **at least one** (quantification existentielle), **at most one**, **exactly one**, etc.

### III.1.3 Ecriture des règles en SBVR

Pour construire une règle en SBVR-SE [OMG, 2008], il faut tout d'abord spécifier les termes métier de la règle candidate et l'ensemble des faits atomiques qui montrent les liens entre ces termes (et éventuellement avec d'autres termes métier ne figurant pas dans la règle candidate). Ensuite, la règle candidate peut être réécrite de façon désambiguïsée et précise en tenant compte de la liste des faits atomiques identifiés et des mots-clés de formulation logique pour les connecter. Et à chaque mot de la règle candidate est associé un style selon son type (termes, relations, mots-clés). Considérons la règle candidate : *An order must not be shipped if the outstanding balance exceeds credit authorization.*[3]. La construction de la règle SBVR-SE pourrait donner ceci (nous donnons les définitions en français) :

3. Une commande ne doit pas être expédiée si le solde dépasse le crédit autorisé.

**Définir le vocabulaire métier** :

| | |
|---|---|
| **Order** | (une commande) |
| *Definition 1* | L'ordre écrit par lequel un client demande à recevoir une liste de marchandises et s'engage à la payer |
| *Definition 2* | Un ensemble de marchandises constitué selon la liste de la définition 1 et destiné à l'auteur de l'ordre |
| **Credit** | |
| *Definition* | Accord pour payer une dette postérieurement au fait générateur |
| **Outstanding balance** | (solde) |
| *Definition* | Le résultat de la compensation entre les avoirs et les dettes d'un client |
| **Credit authorization** | (crédit autorisé) |
| *Definition* | Le montant maximum de dettes cumulées accepté pour un client. |

**Spécifier les faits** :

a customer *places* an order [4]

a customer *has* a credit authorization [5]

a customer *holds* an account [6]

an account *has* an outstanding balance [7]

**Révision contrôlée** :

An order must not *be shipped* if the outstanding balance of the account *held by* the customer that *placed* the order *exceeds* the credit authorization of the customer. [8]

L'utilisation d'un langage contrôlé comme SBVR-SE permet alors de désambiguïser et simplifier les structures syntaxiques complexes de la langue. Ce travail est effectué par l'expert des règles qui est donc amené à apprendre à manipuler le langage contrôlé pour écrire manuellement ses règles candidates. Il n'est pas toujours évident que cette prise en main du langage contrôlé soit effective pour l'analyste. Pour le langage SBVR-SE, à travers leur outil NL2SBVR [9], Bajwa et al. [Bajwa et al., 2011] ont proposé une approche de génération automatique de règles SBVR

4. un client passe une commande
5. un client a un crédit autorisé
6. un client détient un compte
7. un compte a un solde
8. Une commande ne doit être expédiée si le solde du compte détenu par le client qui a passé la commande dépasse le crédit autorisé à un client.
9. http ://www.cs.bham.ac.uk/ isb855/nl2ocl/projects.html

à partir de spécifications des règles produites par un expert en langage naturel. Leur approche de traduction consiste tout d'abord à identifier les termes métier et les faits atomiques de la règle candidate en combinant son arbre syntaxique et une analyse en parties de discours (POS). Puis les termes pleins sont liés à un modèle UML du domaine pour créer le vocabulaire métier de domaine. Ensuite, les faits sont interconnectés les uns aux autres pour reconstruire la règle candidate complète suivant une des notations, SBVR-SE ou RuleSpeak. Pour ce qui concerne la sortie en langage structuré, les connecteurs initiaux sont remplacés par des mots-clés de SBVR-SE. Plus précisément, les opérateurs de quantification "*more than n*" et "*greater than n*" sont remplacés par "*at least n*", "*less than n*" par "*at most n*", "*equal to n*" par "*exactly n*", etc. Les opérateurs modaux "*can*" et "*could*" deviennent "*It is possible*", "*should*" et "*have to*" deviennent "*It is obligatory*". Le traitement de la règle en langage naturel "A person's age must be 18 years" donné en exemple. Elle est traduite en SBVR-SE par "It is obligatory that a person's age should be at least 18 years"

Bajwa et al. [Bajwa et al., 2011] ont testé leur outil sur des phrases à trois niveaux de difficultés. Les deux premiers (règles 'simples' et 'complexes') ne comportent que des règles à fait unique. Les règles complexes sont celles qui utilisent une association entre deux concepts, par exemple « *A vehicle's colour can be white or black* »). Enfin, le dernier niveau dit composé, correspond aux règles composées de plus d'un fait (ex. : « *If a person's salary is more than 10,000£ then the person can buy a car with white colour*»). Ils utilisent un échantillon de 30 règles (10 pour chaque niveau de complexité) extraites du corpus « EU-Rent». C'est un exemple construit qui figure en annexe dans la norme SBVR [OMG, 2008] et qui décrit des règles métier d'un organisme de location de voiture. NL2SBVR traduit correctement plus de 80% des règles. Ces résultats doivent cependant être interprétés prudemment : d'une part la formulation des règles dans le langage source est très simple, d'autre part ni la liaison du vocabulaire au modèle , ni la méthode de validation (il n'existe pas actuellement de validateur SBVR) ne sont précisément décrits.

### III.1.4 Traduction formelle des règles SBVR

L'étape suivante consiste à transposer les règles dans un formalisme à partir duquel elles pourront être implémentées. Les formalismes utilisés ont des propriétés différentes. Par exemple UML fournit un ensemble de représentations graphiques sous forme de diagramme et est considéré comme très intuitif quand il s'agit de spécifier un processus portant sur des objets. Il est facilement utilisé pour décrire le domaine métier et beaucoup moins pour décrire les règles. RIF ou SWRL sont des langages formels de règles qui représentent directement la notion. SWRL

est une combinaison du OWL pour formaliser le vocabulaire métier et du langage RuleML pour formaliser les règles. Il est associé à une définition unique de ses inférences (inspirées de la programmation logique) alors que RIF a plusieurs branches selon que l'inférence est celle de la logique classique, de la programmation logique ou des règles de production. Tous deux disposent d'une syntaxe en balises XML pour l'échange des règles.

Le passage du langage contrôlé au langage formel se fait par une correspondance des éléments linguistiques du langage contrôlé avec un équivalent dans le langage formel. Des travaux ont d'ailleurs été proposés pour la traduction automatique de SBVR en UML/OCL [Bajwa and Lee, 2011] ou de SBVR dans des langages de règles comme SWRL[Ceravolo et al., 2007]. Ces travaux donnent l'impression que, comparée aux difficultés dont nous avons donné un échantillon dans l'introduction, la traduction langage contrôlé-langage formel n'est pas aussi difficile et que toute la difficulté est concentrée dans le passage langage naturel-langage contrôlé. En effet, seules les règles en langage contrôlé ayant une bonne correspondance terme à terme avec le langage formel choisi sont traduites par ces méthodes. Notre conclusion est que les autres nécessitent d'être retravaillées pour être traduites. Autrement dit, pour que la réussite soit atteinte, il faut que le langage contrôlé soit reformulé en langage contrôlé en intégrant les éléments qui permettent la traduction dans un langage de règles (voir IV.1).

Certains de ces éléments sont liés à l'organisation des textes d'où sont extraites les règles. D'autres tiennent à la sémantique de certains éléments de langage contrôlé. Nous avons présenté SBVR (voir IV.1.2.1) comme un bon candidat pour faciliter la traduction, mais la typologie des règles visées doit cependant être prise en compte. Dans [OMG, 2008], une règle métier est dite formellement représentée lorsqu'elle est exprimée uniquement dans les termes métier de domaine qui sont liés les uns aux autres à travers des formulations logiques rendues possibles par des opérateurs logiques, des quantificateurs, etc. Cependant un certain nombre de formulations en langage contrôlé, notamment les formulations modales, ne peuvent être formalisées directement : leur traduction en règle dépend de l'application finale. Nous avons commencé à explorer ces cas de figure [Lévy et al., 2012] pour comprendre quelles connaissances permettent l'opérationnalisation.

## III.2 La simplification textuelle

La prise en compte des textes réglementaires pour l'acquisition des règles soulève une vraie question de simplification des textes par les analystes pour faciliter leur compréhension et leur formalisation. Les langages contrôlés ont été proposés aux analystes afin de contourner les

structures syntaxiques et sémantiques complexes de la langue naturelle. Cependant, comme les langages contrôlés imposent une syntaxe spécifique dans l'écriture des règles qui n'est pas toujours maîtrisée par les experts, les techniques de simplification textuelle peuvent avoir de réels avantages pour aider la traduction du langage naturel en langage contrôlé.

La simplification de texte a fait l'objet de plusieurs travaux mais leurs résultats ne sont pas utilisés dans le monde des règles métier. Ces travaux [Siddharthan and Caius, 2003] portent sur l'analyse syntaxique dans le domaine du traitement automatique de la langue (TALN), l'aide pour les personnes souffrant de troubles de compréhension [Max, 2006], le résumé automatique de texte, l'apprentissage de la langue, etc. Les travaux s'efforcent de simplifier la complexité de la langue naturelle pour résoudre les difficultés de compréhension que peut présenter un texte sans remettre en cause son sens et les informations qu'il contient.

Dans les textes réglementaires une règle métier est transcrite dans une unité textuelle, le plus souvent une phrase [10]. Les travaux de simplification quant à eux s'accordent [Gasperin et al., 2009], [Siddharthan and Caius, 2003, Chandrasekar et al., 1996, Chandrasekar and Srinivas, 1997] à dire que pour simplifier un texte il faut simplifier les phrases qui le composent (simplification syntaxique) et simplifier les mots utilisés dans ces phrases (simplification lexicale). Il est légitime d'étudier l'apport des méthodes de simplification à notre objectif de travail sur des textes réglementaires.

### III.2.1 Simplification lexicale

La simplification lexicale consiste à réduire la complexité lexicale d'un texte tout en préservant son sens et les informations qu'il contient [Siddharthan and Caius, 2003]. En effet, puisque le sens d'un texte est englobé en grande partie dans les termes utilisés, il s'agit alors de les simplifier en les remplaçant par des termes plus simples et facilement compréhensibles [De Belder et al., 2010]. Ce sont en général des synonymes pouvant être tirés d'une base lexicale (par exemple un dictionnaire) ou générés à partir du texte lui-même (ou d'un corpus différent) suivant des modèles LWLM (Latent Words Language Model) [De Belder et al., 2010].

### III.2.2 Simplification syntaxique

La simplification syntaxique est la plus utilisée [Siddharthan, 2002, Carroll et al., 1998, Chandrasekar et al., 1996]. Elle consiste à réduire la complexité syntaxique des phrases d'un texte tout en préservant son sens et les informations qu'il contient [Siddharthan and Caius, 2003].

10. http ://www.omg.org

Le but de la simplification syntaxique est de simplifier toutes les phrases complexes du texte. Selon le Dr Spielmann [11] [12] « une phrase dans sa forme la plus élémentaire, correspond à une proposition (appelée phrase simple) constituée par l'association d'un sujet (ce dont on parle) et d'un prédicat (ce qu'on dit du sujet) qui inclut généralement un verbe et ses compléments ». Une phrase complexe quant à elle correspond « à plusieurs propositions liées entre elles ». Ces liens entre phrases simples qui permettent la construction d'une phrase complexe sont des phénomènes de la langue identifiés dans les travaux de simplification de texte [Siddharthan, 2002, Candido et al., 2009]. Selon eux il existe cinq types de phénomènes :

- La juxtaposition qui est un lien implicite permettant d'unir deux propositions par des signes de ponctuation (virgule, point-virgule, deux points, etc.).
- La coordination qui est un lien explicite exprimant une relation entre deux propositions. Le lien est défini suivant des connecteurs tels que des conjonctions de coordination (« for », « and », « nor », « but », « or », « yet », « so »pour exprimer l'opposition, l'addition, l'alternative, la conséquence, la cause, l'explication ou la justification) ou des adverbes (« also », « in fact », « indeed », « then », « as well as », « next », etc pour exprimer des relations logiques ou la progression).
- La subordination qui exprime de façon explicite une relation de dépendance entre deux propositions. Une proposition subordonnée est ainsi liée à une proposition principale soit par un pronom relatif (« who », « which », « where », « whose »), soit par une conjonction de subordination (« as », « as soon as », « when », « since », « if », « so that », « because », « whereas »pour exprimer le but, la condition, l'hypothèse, la cause, la conséquence, la restriction, etc), soit par une préposition (« as a result of », « as for », « as from », « as opposed to », « except for », « in addition to »).
- La voix passive, qui contrairement à la voix active (où le sujet fait l'action), exprime une forme où le sujet subit l'action. La voix passive se construit uniquement sur des verbes transitifs possédant un complément d'objet direct (COD). Elle est introduit par l'auxiliaire « be » (qui est conjugué au même temps que le verbe à la voix active) et le mot « by » introduisant un complément d'agent (qui est le sujet à la voix active).
- Le désordre des éléments d'une phrase (mots, termes, expressions, propositions, etc.) qui crée aussi une certaine forme de complexité de la phrase. Une phrase simple est sous forme de sujet-verbe-complément et donc lorsque cet ordre n'est pas respecté la phrase devient plus difficile à comprendre. Le désordre peut aussi être incriminé dans le cas où deux propositions sont ordonnées de façon inversée. Par exemple dans le cas d'une

11. https ://www9.georgetown.edu/faculty/spielmag/
12. http ://www9.georgetown.edu/faculty/spielmag/docs/txt/laphrase.htm

subordination exprimant une condition (if-then), la prémisse peut être précédée de la conclusion.

Pour simplifier ces phénomènes complexes de la langue, Candido et al. [Candido et al., 2009] ont proposé trois types d'opérations de structuration syntaxique pouvant s'appliquer sur des phrases présentant ces complexités :

- Le découpage de phrase est une des opérations les plus fréquentes. Elle consiste à subdiviser une phrase initiale en plusieurs phrases simples à partir des marqueurs de liens tels que les signes de ponctuation marquant le phénomène de la juxtaposition, les prénoms relatifs, les adverbes, les propositions, les conjonctions de coordination ou de subordination.
- La transformation à la voix active est une opération s'appliquant sur une phrase à la voix passive.
- L'ordonnancement s'applique pour restructurer la phrase suivant le triplet sujet-verbe-complément ou tout simplement pour réordonner les propositions simples qui font la phrase complexe dans le cas d'un phénomène d'inversion.

### III.2.3 Les techniques de simplification

Le processus de simplification de texte tel que présenté dans les travaux de Carroll et al. [Carroll et al., 1998] et de Siddharthan et al. [Siddharthan, 2002] repose sur une architecture en deux phases principales.

Une première phase d'analyse correspondant à une phase de prétraitement du texte à simplifier. Cette phase fournit une représentation structurée de chaque phrase du texte afin de les préparer pour la procédure de simplification à proprement parler. L'analyse correspond au lancement d'outils TAL en particulier des analyseurs syntaxiques sur chacune des phrases. A la suite de l'analyse, chaque phrase est représentée sous forme d'un arbre syntaxique [13] étiquetant chaque mot avec sa catégorie syntaxique.

La seconde phase correspond à la phase de transformation. Autrement dit, il s'agit de la procédure de simplification qui peut être lexicale ou syntaxique.

Pour la simplification lexicale [De Belder et al., 2010], il s'agit, à partir de l'arbre syntaxique correspondant à une phrase, d'analyser la morphologie de chaque mot permettant de le décomposer en son radical et sa flexion (préfixe, suffixe, etc.). Ces informations permettent d'identifier le lemme. Un ensemble de synonymes sont extraits d'une base lexicale, par exemple WordNet [14] dans [Carroll et al., 1998]. Tous les synonymes sont évalués et le plus simple est choisi et asso-

13. http ://en.wikipedia.org/wiki/Tree-adjoining_grammar
14. http ://fr.wikipedia.org/wiki/WordNet

cié à la flexion du radical d'origine. Cette simplification peut être effectuée manuellement mais généralement des algorithmes automatiques sont définis pour cela. Ces algorithmes utilisent des outils TAL pour déterminer le lemme (qui est parfois même fourni par l'outil d'analyse syntaxique) d'un mot de l'arbre syntaxique, avant de faire une correspondance automatique entre le mot et le dictionnaire pour déterminer les synonymes du mot.

La simplification syntaxique [Chandrasekar et al., 1996, Chandrasekar and Srinivas, 1997] quant à elle consiste à appliquer des patrons linguistiques (appelés aussi règles de transformations) qui sont exécutés sur les arbres syntaxiques des phrases. Chaque patron spécifie une prémisse qui détermine quelle condition doit remplir une phrase pour faire l'objet d'une simplification (c'est-à-dire quand une phrase présente un phénomène complexe). La conclusion du patron permet alors de dire quelle opération de simplification il faut pour simplifier ce phénomène. Les patrons peuvent être définies de façon manuelle [Chandrasekar et al., 1996], ou être appris automatiquement [Chandrasekar and Srinivas, 1997] à partir d'un corpus textuel test. Ce corpus répertorie des exemples de phrases reflétant des phénomènes complexes ainsi que leurs simplifications respectives. Ensuite des outils d'apprentissage automatiques prennent ces couples pour générer des patrons linguistiques qui peuvent alors être utilisés pour simplifier d'autres phrases présentant des phénomènes complexes. Ces mêmes outils se chargent d'appliquer un patron sur l'arbre syntaxique d'une phrase complexe.

## III.3 L'annotation sémantique

Dans les travaux de formalisation de règles, le texte est utilisé essentiellement comme source d'information. Lorsque les règles candidates sont identifiées, on ne parle plus du texte d'origine. Et pourtant, le texte est une ressource très précieuse. Nous avons vu (II.1.4) que la maintenance des règles est un objectif important du SGRM. Cela inclut la mise à jour des règles lorsque les besoins du système d'information évoluent. Cette évolution est répercutée dans les textes de base par les experts avant d'être prise en compte dans le système. Il est donc important de garder un lien entre le texte et les règles et qu'ainsi à chaque fois qu'il faudra modifier une règle il suffira d'observer les modifications sur le fragment textuel (la phrase) d'où elle dérive. Mieux encore, le texte peut être utilisé pour vérifier la cohérence des règles extraites : il est important que les règles intégrées dans le SGRM soient conformes aux besoins initialement définis dans le texte. Il peut même aussi servir à justifier les décisions du SGRM.

Cette dimension n'est pas souvent prise en compte dans les travaux sur l'acquisition des règles. Pour maintenir les règles par rapport aux besoins du système d'information, la solution

qui est souvent proposée est de retraduire les règles formelles du SGRM en langage naturel [Cabot et al., 2010] afin qu'elles puissent être modifiées par les experts. Si le texte est partie intégrante du SGRM, il devient inutile de faire cette retraduction. Des travaux ont été proposés pour articuler un texte en langue naturelle à un modèle formel, notamment les travaux sur l'annotation sémantique et sur l'indexation de documents. Ces travaux fournissent aussi des techniques d'analyse et d'exploration sémantique du texte qui facilitent la modélisation des connaissances enfouies dans le texte et qui peuvent être très utiles pour une meilleure exploitation et compréhension du texte réglementaire.

### III.3.1 Annoter pour indexer

Le terme d'« annotation » désigne, en général, un apport d'informations de nature interprétative à un texte [Leech, 1997]. On parle alors d'annotation sémantique pour désigner la définition d'une sur-couche d'informations sémantiques qui précisent ou complètent le sens du texte [Leech, 1997]. L'annotation sémantique a été introduite dans divers travaux d'analyse de corpus textuel. Elle facilite la manipulation d'un texte en favorisant son analyse et son exploration sémantique, et en assistant l'acquisition [Amardeilh et al., 2005] et la modélisation [Uren et al., 2006] des connaissances qu'il contient.

Concrètement, annoter un texte sémantiquement consiste à le connecter à des informations sémantiques. Celles-ci peuvent être de simples notes (ou commentaires) apposées par un lecteur qui s'intéresse à certains passages du texte ou aussi des tags posés par les usagers du Web 2.0 [Oren et al., 2006]. Dans le Web sémantique (où le terme est très utilisé), ces informations sémantiques sont regroupées dans des modèles sémantiques formels représentés le plus souvent dans des ontologies. Pour cette raison, Amardeilh [Amardeilh, 2007] définit (dans sa thèse) l'annotation sémantique comme étant une représentation formelle d'un contenu, exprimé à l'aide de concepts, de relations et d'instances, et reliée à un texte.

Comme nous l'avions dit dans [Guissé et al., 2009], l'annotation sémantique peut être utilisées soit pour la valeur sémantique qu'elle représente soit pour l'accès au texte qui lui a donné naissance. Dans le premier cas, le texte sert uniquement de source d'information pour l'acquisition et la modélisation de connaissances. Dans le second cas, la ressource sémantique servant à annoter le texte est utilisé pour indexer le texte. Cette deuxième dimension de l'annotation sémantique est appelée indexation [Guissé et al., 2009, Prié and Garlatti, 2000, Sidhom et al., 2005, Lévy et al., 2010] et c'est ce cas d'usage de l'annotation qui nous intéresse ici.

De façon générale, le terme d'« indexation sémantique » renvoie à une indexation concep-

tuelle d'un texte qui consiste à rattacher un concept à chaque terme du texte (mots ou groupe de mots) [Voss et al., 1999]. Ce terme traduit une forme d'indexation qui ne se limite plus à un accès simple à un texte suivant les mots qu'il contient mais qui s'intéresse aussi au contenu sémantique du texte [Prié and Garlatti, 2000]. Pour Lévy et al. [Lévy et al., 2010], l'indexation sémantique est inséparable de l'annotation : l'indexation sémantique résulte du processus qui consiste à poser des annotations sémantiques sur un texte et à les rassembler suivant une organisation sémantique. Et donc, l'indexation ne peut se faire sans la description formelle des connaissances sémantiques du texte qui doivent être définies avant l'indexation. En plus de cette description, il est aussi nécessaire d'établir la correspondance entre entités sémantiques et fragments textuels. Les entités sémantiques sont en général décrites dans un vocabulaire de domaine rassemblé dans un dictionnaire, un thésaurus ou une ontologie. L'indexation repose alors sur une projection de cette description sémantique sur le texte. Ma et al. dans [Ma et al., 2009, Ma et al., 2010] proposent la notion d'*ontologie étendue*, dans laquelle la description formelle de l'ontologie est étendue par une description du vocabulaire et des règles de projection ; ces dernières permettant de résoudre les ambiguïtés sémantiques du vocabulaire en fonction du contexte.

Cette forme d'indexation est très utilisée dans les systèmes de recherche d'information notamment dans les systèmes à bases d'ontologie [Braga et al., 2000]. Elle est utilisée pour une forme de recherche [Kiryakov et al., 2004] axée sur une exploration sémantique d'un texte à partir de la structure sémantique qui décrit son contenu. Les annotations sémantiques liant ce texte et sa structure sémantique constituent un espace de navigation [Guissé et al., 2009]. Celui-ci articule les textes et leur structure sémantique, permettant ainsi l'accès à un texte ou à un fragment de texte à partir des concepts qu'il contient et vice-versa. Cet espace permet en outre de faire des mécanismes d'inférences et de raisonnement en vue d'une recherche d'information sémantique. Du texte à la structure sémantique, l'indexation est aussi utilisée pour assister la construction [Aussenac-Gilles et al., 2008] et la population d'ontologie [Amardeilh et al., 2005], mais aussi dans l'exploration et l'interrogation sémantique de corpus textuel [Widlocher and Mathet, 2009, Abacha and Zweigenbaum, 2010]. Dans le sens contraire, l'indexation sert, vis-à-vis du texte, à tracer et documenter les ressources sémantiques si celles-ci ont été acquises à partir du texte ou à les maintenir quand le texte évolue.

### III.3.2 Représentation formelle des annotations

La représentation formelle d'un index correspond à la fois à la description formelle des modèles documentaire et sémantique mais aussi des annotations sémantiques qui les lient. Les sys-

tèmes d'indexation conceptuelle ne perdent pas de vue qu'il est nécessaire que les formalismes de représentation utilisés puissent s'entréchanger et soient compatibles [Prié and Garlatti, 2000]. Autrement dit, il est par exemple nécessaire que les éléments de langage du formalisme représentatif du modèle sémantique puissent s'intégrer à la structure formelle des documents à annoter.

#### III.3.2.1 Technologies du Web sémantique

Plusieurs langages de représentation de la connaissance et d'annotation sémantique ont été proposés (plus de détails dans [Amardeilh, 2007, Oren et al., 2006]). La plupart de ces langages sont essentiellement apparus dans le Web Sémantique et sont surtout standardisés et recommandés par W3C [15] [16]. Ils ont la particularité d'être munis de sémantique formelle permettant d'être compatible les uns les autres [Baget et al., 2005]. Nous nous intéressons particulièrement ici aux langages (Figure III.2) d'assertion ou d'annotation (RDF), de définition de ressources sémantiques (RDFs, OWL), de manipulation et d'interrogation des annotations (SPARQL).

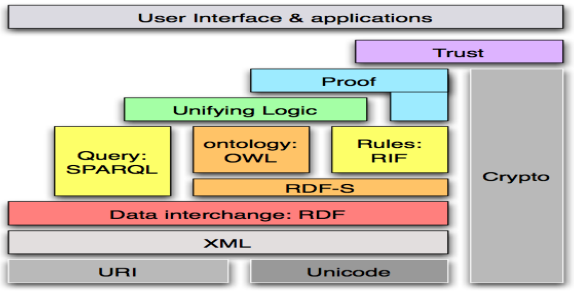



**Figure III.2** – Architecture du Web Sémantique par Berners-Lee [17].

– **RDF** [Klyne and Carroll, 2004, Hayes, 2004](Resource Description Framework) est un modèle formel qui permet de définir des relations entre des ressources du Web (pages web, documents textuels bruts (pdf, txt, doc, etc.), multimédias (images, vidéos), etc.) anonymes ou identifiées par une URI [18]. RDF repose sur un modèle de graphe décrit sous forme de triplets. Chaque triplet correspond à un ensemble de trois éléments : <*sujet-prédicat-objet*> où le sujet représente la ressource à décrire, le prédicat correspond la propriété descriptive du sujet, et l'objet représente la valeur de cette propriété (qui est

15. http ://www.w3.org/
16. http ://fr.wikipedia.org/wiki/World_Wide_Web_Consortium
17. http ://www.w3.org/2009/Talks/0204-campus-party-tbl
18. Uniform Resource Identifier(http ://fr.wikipedia.org/wiki/Identifiants_uniformisés_de_ressource)

soit une valeur littérale (numérique, chaîne de caractères) soit une autre ressource du Web).

RDF est surtout utilisé pour annoter sémantiquement un document dans son ensemble. Les annotations peuvent être des notes, des commentaires, des explications qu'on associe à un document. Elles peuvent aussi être conceptuelle c'est-à-dire qu'elles lient un concept sémantique tiré d'une ontologie afin d'établir le sujet principal du document (par exemple un document qui parle des matchs de la ligue 1 parle de « Football »). Ce type d'annotation est implémenté dans des outils comme SMORE [Kalyanpur et al., 2003] ou Annotea [Kahan et al., 2001]. L'annotation du contenu à proprement parler d'un document consiste le plus souvent à annoter de façon précise des parties, des mots ou des groupe de mots du document. Ce dernier est le cas qui apparaît le plus dans la littérature et est souvent assimilé à l'indexation conceptuelle. Les mots du document sont annotés par des concepts tirés d'un modèle sémantique pouvant par exemple être une ontologie. Des outils ont été proposés pour annoter un document au regard d'un tel modèle sémantique. On peut citer les outils Semtag [Dill et al., 2003] ou KIM [Popov et al., 2003], reposant eux-mêmes sur des outils d'extraction pour identifier les fragments de texte à annoter et leur associer une étiquette sémantique. Il s'agit d'ordinaire de systèmes d'extraction d'informations dans des ressources non structurées qui exploitent une analyse linguistique du texte comme certains modules d'extraction d'information intégrés à la plateforme linguistique GATE [Cunningham, 2002] ou Amilcar [Ciravegna and Wilks, 2003].

– **RDFa** [Adida and Birbeck, 2008, Adida et al., 2008] (RDF in attributes). Dans la majorité de ces outils, les annotations ne sont pas intégrées dans le document, en particulier lorsque celui-ci est non structuré. Elles sont alors stockées dans un document indépendant de triplets RDF. Un tel document serait alors composé d'un ensemble d'occurrences (chacune correspondant à un triplet) d'objets conceptuels (concept, relation ou propriété, ou instance). L'exemple III.3 décrit un ensemble de triplets RDF qui définissent une occurrence d'une instance du concept « Doctorant »(via le prédicat « rdf :type ») ainsi que des propriétés caractéristiques de celle-ci («  skos :prefLabel», « lieu_naissance », « date_naissance »). Récemment (en 2008), le modèle de représentation **RDFa** a été recommandé par le W3C pour intégrer des annotations sous forme de triplets RDF dans le document. Il permet alors d'encapsuler des triplets RDF dans des pages XHTML [19] ; celui-ci respectant la syntaxe XML est compatible avec RDF. Et dans le document, tout fragment de texte annoté correspond à un ou plusieurs triplets RDF où chacun d'eux peut

19. http ://fr.wikipedia.org/wiki/Extensible_HyperText_Markup_Language

aussi servir à annoter un sous-fragment du fragment annoté. L'exemple III.4 reprend ainsi l'annotation RDF de la figure III.3 au format RDFa.

```
<rdf:Description rdf:about="http://www-lipn.univ-paris13.fr#Guisse">
  <skos:prefLabel>Abdoulaye Guisse</skos:prefLabel>
  <rdf:type rdf:resource="#Doctorant"/>
  <lieu_naissance rdf:resource="#Dakar"/>
  <date_naissance>18 Juin 1983</date_naissance>
</rdf:Description>
```

**Figure III.3** – Exemple de triplets décrivant une annotation sémantique d'un fragment de texte.

<html xmlns="http ://www.w3.org/1999/xhtml">
...

L'étudiant *Abdoulaye Guissé* suit sa thèse à Paris 13. Il est né à Dakar le 18 Juin 1983 ...

...
</html>

**Figure III.4** – Exemple d'annotation RDFa.

– **RDFS** [Antoniou et al., 2005] (RDF Schema) a été proposé comme extension du langage RDF pour décrire les ressources sémantiques utilisées pour annoter un document RDF. Il permet d'étiqueter les concepts sous forme de classes afin de structurer les annotations RDF. RDFS est un langage de triplet comme RDF mais destiné à définir le vocabulaire des informations stockées dans les triplets RDF. A travers le vocabulaire, il peut être utilisé pour définir des ontologies très basiques correspondant à une taxonomie de classes et de leurs relations. Cela offre une expressivité sémantique intéressante aux annotations RDF où chaque sujet de triplet RDF est une instance d'une classe conceptuelle du schéma RDF utilisé. Sur la figure III.5, nous montrons des triplets RDFS définissant un petit vocabulaire pour l'annotation précédente (Figure III.3).

– **OWL** [Antoniou et al., 2005] (Web Ontology Language). Les ontologies sont de plus en plus utilisées pour les besoins d'indexation conceptuelle ou d'annotation sémantique de

<**Doctorant** rdf :type rdfs :Class>, le concept **Doctorant** est une classe ;
<**étudiant** rdf :type rdfs :Class>, le concept **étudiant** est une classe ;
<**Doctorant** rdfs :subClassOf **étudiant**>, un **Doctorant** est un **étudiant** ;
<**Ville** rdf :type rdfs :Class>, le concept **Ville** est une classe ;
<**Dakar** rdf :type **Ville**>, **Dakar** est une **Ville**, c'est une instance du concept ;
<**lieu_naissance** rdf :type rdfs :Property>, **lieu_naissance** est une propriété ;
<**lieu_naissance** rdfs :domain **étudiant**>, toute ressource utilisée comme sujet au prédicat **lieu_naissance** est de type **étudiant** (ex :**Doctorant**) ;
<**lieu_naissance** rdfs :range **Ville**>, toute ressource utilisée comme objet au prédicat **lieu_naissance** est de type **Ville** (ex :Dakar) ;
...

**Figure III.5** – Exemple de triplets RDFS.

documents. Une ontologie définit un modèle de données représentatif d'un ensemble de concepts d'un domaine particulier et les relations qu'ils entretiennent. Nous avons vu que RDFS pouvait servir à définir une ontologie simple à partir d'une taxonomie de classes. Cependant RDFS présente des limites qui bornent l'expressivité sémantique des connaissances notamment son impossibilité à raisonner sur celles-ci. Comme une ontologie repose sur un modèle de graphe d'organisation des connaissances où les concepts sont définis les uns par rapport aux autres autorisant le raisonnement sur les connaissances, la définition d'un modèle de représentation ontologique est plus que nécessaire. C'est ainsi que le langage **OWL** (Web Ontology Language) a été proposé par le W3C et permettant de décrire des ontologies. OWL est fondé sur la syntaxe RDF et est une sorte d'extension de RDFS inspiré de la logique descriptive [20]. En effet, il a la capacité de décrire les concepts en classes via une taxonomie comme dans RDFS, mais aussi il propose un ensemble de relations logiques entre les classes et leurs propriétés. Il s'agit notamment de relation de comparaison : identité, équivalence, contraire, cardinalité, symétrie, transitivité disjonction, etc. Il permet aussi d'apposer des contraintes notamment d'intégrité ou des restrictions sur les classes. Tout cela lui confère ainsi une forte expressivité supplémentaire comparer au RDFS et une sémantique formelle permettant de raisonner et mieux manipuler les connaissances. OWL est un langage riche et complexe à la fois, il est défini en trois niveaux : 1) OWL Lite est l'équivalent du RDFS permettant de définir une taxonomie de classes. 2) OWL DL est équivalent à une logique de description assez complexe mais décidable. Il offre la possibilité d'inférer et d'effectuer des calculs logiques sur les connaissances. 3) OWL Full est une extension de OWL DL autorisant la représentation de collections de classes et laissant la possibilité qu'une classe soit instance d'une

20. http ://fr.wikipedia.org/wiki/Logique_de_description

autre classe. La logique de description sous-jacente n'est pas décidable.

- **SKOS** [Miles et al., 2005, Assem et al., 2006] (Simple Knowledge Organization System). Basé aussi sur le modèle RDF, SKOS est utilisé pour décrire des terminologies telles que des vocabulaires, des thésaurus, des dictionnaires, des glossaires, etc. De ce fait, il est souvent combiné aux modèles de type OWL afin de définir des ontologies lexicalisées c'est-à-dire des ontologies enrichies de connaissances lexicales (terminologiques) [Buitelaar et al., 2011]. C'est ce type d'ontologies qui est dans la méthode de construction TERMINAE [Aussenac-Gilles et al., 2008] où les concepts ontologiques sont associés à un ensemble de valeurs terminologiques synonymes. Dans ce cas précis, SKOS permet d'associer aux concepts des termes préférentiels et des termes alternatifs qui dotent une ontologie d'une sémantique linguistique, laquelle sert ensuite à améliorer l'annotation sémantique d'un contenu textuel[Bechhofer et al., 2002].
- **SPARQL**[21] (SPARQL Protocol and RDF Query Language) est un langage de requête inspiré du SQL et essentiellement proposé pour manipuler des données RDF. Il permet de rechercher, ajouter, modifier ou supprimer des données dans un graphe RDF. Il est aussi utilisé pour naviguer dans un ensemble d'annotations RDF, dans des vocabulaires RDFS ou OWL ou pour naviguer entre les annotations et les vocabulaires.

#### III.3.2.2 Manipulation de graphes RDF

L'indexation sémantique qui consiste à projeter une ressource sémantique sur le ou les documents à annoter [Lévy et al., 2010], définit un espace de liens entre ces deux types de connaissances. Toute l'exploitation de l'index construit repose sur l'exploitation de cet espace de liens. Elle permet de naviguer entre les ressources « annotateurs » et celles annotées notamment entre les documents et le modèle sémantique. Cette exploitation est le plus utilisée en recherche d'information [Prié and Garlatti, 2000] car les liens permettent d'articuler ces deux types de ressources c.-à-d. qu'ils permettent d'identifier tous les éléments sémantiques qui ont servi à annoter un document ou une partie d'un document, ou vice versa tous les documents annotés par un élément sémantique donné. Celui-ci, si le modèle sémantique utilisé est une ontologie, correspond soit à un concept de domaine, à une relation conceptuelle ou à une instance de concept.

Nous nous intéressons à l'exploitation de ces liens, qui peuvent être des ressources précieuses pour assister l'acquisition et la maintenance des règles notamment pour améliorer l'analyse et l'exploration sémantique des textes. L'exploitation de ces liens est fonction des formalismes

21. http ://www.w3.org/TR/rdf-sparql-query

utilisés pour la représentation du modèle d'index. La plupart de ces formalismes reposent sur des modèles de graphes qui leur confèrent les mêmes propriétés structurelles et logiques ainsi qu'une réelle puissance de raisonnement et de manipulation. Les technologies du Web sémantique décrites précédemment utilisent aussi des représentations sous forme de graphes comme le montre F. Gandon de façon détaillée dans son mémoire d'HDR [Gandon, 2008] dans lequel il soutient que l'usage de « la structure de graphe offre des opportunités d'optimisation de l'indexation et de l'organisation des connaissances ainsi qu'un espace permettant de définir des inférences dépassant le raisonnement logique ». Selon lui, cette structure fournit un espace formel d'échange et de manipulation des connaissances. Nous ne plongerons pas dans une étude du caractère graphique des formalismes de représentation des annotations et de l'index mais nous nous focalisons sur ce que son exploitation apporte à l'acquisition et la maintenance des règles (voir V.3 et V.4).

Ces technologies sont définies suivant une architecture en couches (voir III.2) dont la base est le formalisme RDF. Celui-ci utilise une structure XML et permet d'organiser les données sous forme de triplet (*objet-prédicat-valeur*) suivant un graphe étiqueté dont les sommets sont les ressources et les valeurs et dont les arêtes sont les assertions étiquetées par les prédicats. RDF [Bizer et al., 2007] est un format servant à interconnecter des données provenant de plusieurs sources différentes. Dans un contexte d'indexation conceptuelle, les données correspondent aux documents et aux éléments de la ressource sémantique utilisée pour l'annotation, soient les concepts si celle-ci est une ontologie. L'interconnexion [Tauberer, 2008] s'effectue sur la base des triplets : un objet est connecté à une valeur suivant un prédicat donné, cette valeur peut aussi être un objet qui a son tour peut être connecté à une autre valeur par un autre prédicat, ainsi de suite, pour former au final un vrai graphe conceptuel [22]. Et pour arriver à une organisation et une structuration cohérentes, RDF associe à chaque nœud du graphe une URI unique notamment aux objets, aux prédicats et aux valeurs (voir III.6).

RDF [Klyne and Carroll, 2004, Hayes, 2004] sert de format d'échange pour les formalismes représentatifs des annotations, des ressources annotées, et des ressources sémantiques. En effet, de la même manière que XML est un métalangage permettant de décrire tous les langages XML, RDF sert aussi de base à la pile de formalismes du Web sémantique (voir III.2). RDF fournit un cadre commun aux formalismes comme RDFs, RDFa, OWL, SKOS. Ainsi, l'utilisation conjointe de ces derniers est essentiellement rendue possible par les fonctionnalités que peut offrir un graphe RDF.

Les fonctionnalités d'un graphe RDF reposent sur des requêtes [Angles et al., 2004] qui dé-

22. http ://www.rdfabout.com/
23. http ://fr.wikipedia.org/wiki/Resource_Description_Framework

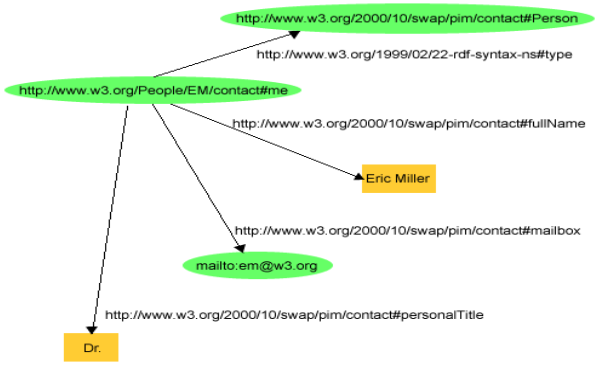



**Figure III.6** – Exemple de graphe RDF [23].

crivent des opérations susceptibles d'être exécutées sur un tel graphe, et qui permettent un système de navigation dans l'espace de liens entre les documents et les ressources sémantiques. Plusieurs usages de ces fonctionnalités existent dans la littérature [Hayes and Gutierrez, 2004] mais l'usage le plus connu est celui lié au cadre Web de données ("linked data", en anglais) [Bizer et al., 2007, Bizer et al., 2009] qui propose des techniques plus avancées et intelligentes d'exploration des ressources du Web. Pour mettre en œuvre ces requêtes sur un graphe RDF plusieurs langages de requêtes [Haase et al., 2004] (SPARQL, RQL, Triple, etc) ainsi que des moteurs sémantiques [Gandon, 2008] (CORESE [Corby et al., 2004], JENA, Sesame, etc.) permettent de stocker le graphe et d'y appliquer les requêtes. Il existe plusieurs types de requêtes élémentaires [Angles et al., 2004] utilisés pour construire la navigation dans un graphe RDF :

- recherche de nœuds adjacents. Ces requêtes permettent d'articuler les ressources en naviguant entre les nœuds des triplets RDF. Elles permettent entre autre de retrouver toutes ressources adjacentes à une ressource donnée. Pour exemple, dans un contexte d'indexation conceptuelle, elles permettent de retrouver tous les documents annotés par un concept ontologique donné ou vice versa ;
- recherche de liens adjacents. Ces requêtes permettent de naviguer dans le graphe RDF de liens (ou prédicats) afin de retrouver tous les liens menés ("domain") par une ressource donnée ou impliquant une ressource donnée ("range"), partagés par des ressources différentes, etc. Ce type de requêtes permet de d'identifier toutes les informations adjacentes à une ressource donnée ;
- conduite de calculs. Ce sont des requêtes qui correspondent à des opérations de calcul sur le graphe. Elles permettent de déterminer par exemple le nombre de ressources qu'implique un lien donné ou plus pratiquement le nombre de documents ayant été annotés par un concept ou vice versa.

– recherche de ressources similaires. Ces requêtes permettent de déterminer les similitudes entre les ressources. Elles permettent de retrouver toutes les ressources ayant les mêmes caractéristiques communes. Par exemple tous les documents annotés par les mêmes concepts.

# Deuxième partie

# Notre démarche

# Chapitre IV

# Cadre général

## IV.1 Analyse des besoins

Notre travail porte sur la question de l'acquisition des règles à partir de ressources textuelles. Nous avons repris de l'état de l'art (voir II.2) la division en trois étapes successives, une étape d'identification des règles à partir des textes, une étape de formalisation des règles en vue de leur automatisation et enfin une phase de maintenance des règles une fois formalisées. Plusieurs techniques, outils et méthodologies ont été proposés pour la gestion de chacune de ces étapes. Nous intéressons d'abord particulièrement aux techniques les plus utilisées, les plus adaptées et les plus expérimentées afin d'aider au mieux à acquérir les règles.

### IV.1.1 Un besoin d'identifier les règles

Les travaux d'identification des règles à partir de textes obtiennent des résultats (voir II.2.2.1) très satisfaisants sur un certain type de textes réglementaires. Il s'agit de textes où les règles sont très visibles et dans lesquels les mots-clés caractéristiques de règles sont bien représentés. Dans ces cas l'application des patrons linguistiques de détection permet d'extraire plus de règles et plus de règles métier valides (une précision de 71% et un rappel de 72% (voir II.2.2.1)). Ces textes sont généralement très artificiels, chaque règle étant déja isolée dans une phrase avec des tructures syntaxiques simples. Le texte EU-Rent [1] est un bon exemple de texte artificiel.

Toutefois, ces résultats sont difficilement généralisables sur l'ensemble des textes réglementaires. En effet, beaucoup de textes sont plus axés sur les besoins de communication dans l'entreprise et au delà ; ils sont très souvent rédigés avec l'ambiguîté inhérente à la langue na-

1. http ://www.businessrulesgroup.org/first_paper/br01ad.htm

turelle mais surtout avec un style juridique marqué qui rend difficile l'identification des règles métier. La détection automatique suivant des patrons linguistiques n'est alors pas adaptée. Elle nécessite une intervention humaine plus importante. Les experts chargés de cette tâche sont plus dans la compréhension et l'exploration sémantique des textes afin d'identifier à la main les règles. Mais un travail manuel est difficilement réalisable sur des textes de grande taille parce qu'il coûte très cher en temps. Dans ce cas, la fouille linguistique, à base de patrons ou autre, peut être utile notamment pour filtrer automatiquement les phrases pertinentes avant de faire appel à la touche humaine, celle de l'expert pour valider les phrases pertinentes qui sont des règles métier.

### IV.1.2 Un besoin de formaliser les règles

Le texte une fois identifié doit être formalisé. Notre but est d'outiller cette étape, de l'appuyer sur les méthodes du TAL. D'une part, ces méthodes restent dans le domaine du langage naturel ; d'autre part, il n'y a pas de langage formel commun, même abstrait, qui puisse être une cible indépendamment de la famille de règles choisie : règles de production, programmation logique ou logique classique (cf. par ex. l'exposé de Ivan Herman à IC 2008 [2]). Nous avons donc pris pour cible des transformations un langage contrôlé, sachant que ces langages sont déjà utilisés comme source dans des processus de formalisation plus faciles à mettre en œuvre (cf. par exemple [Dubauskaite and Vasilecas, 2009]). Nous avons choisi SBVR-SE pour jouer ce rôle de pivot, parce que ce langage vise explicitement les règles métier, parce qu'il est l'objet d'une norme de l'OMG, et parce qu'un spécialiste de ce langage participait à ONTORULE [3]. Il s'agit de réduire les difficultés d'interprétation de la langue naturelle. Celles-ci peuvent être prises en compte à travers des techniques de simplification textuelle inspirées des travaux étudiés en III.2. Nous nous focalisons ici uniquement sur le passage de la langue naturelle au langage contrôlé en laissant de coté celui du langage contrôlé au langage formel (voir IV.2).

#### IV.1.2.1 SBVR-SE comme langage contrôlé intermédiaire

La principale utilisation que nous attendons de SBVR-SE est qu'il puisse offrir une expressivité qui permette de traduire les règles en langage naturel de façon fidèle tout en réduisant les difficultés d'interprétation. On peut décrire, à partir des travaux étudiés en II.2.2 et des

2. `http://www.w3.org/2008/Talks/0619-Nancy-IH/`
3. Nous remercions John Hall pour sa disponibilité tout au long du projet, et pour la patience avec laquelle il a expliqué SBVR et son utilisation par les experts métier

exemples d'emploi de SBVR fournis avec la norme, une méthodologie experte de transformation en langage contrôlé. L'écriture commence par une spécification sémantique de la règle. Chaque terme de la règle et les relations le liant avec les autres termes de la règles sont explicités à partir du vocabulaire métier. L'étape suivante consiste à déterminer le fait principal (sujet-verbe-complément) qui concentre tout le sens de la règle. Une règle est écrite sous forme de phrase en SBVR-SE, elle contient au moins un fait principal (ou noyau de la phrase) et peut ou pas disposer de faits secondaires. Une fois le fait principal déterminé, des opérateurs modaux (obligation, nécessité, impossibilité,) ou de quantification (the, each, at least n,...) peuvent être appliqués sur le sujet ou sur les compléments de ce fait afin de clarifier leur valeur sémantique. Ensuite, tous les autres faits secondaires sont interconnectés avec le fait principal à l'aide d'opérateurs logiques (and, or, not). Ainsi, SBVR-SE fournit tous les outils (Figure IV.1) nécessaires pour exprimer fidèlement la règle.

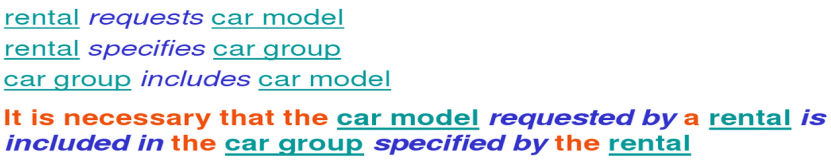



**Figure IV.1** – Exemple de règle SBVR autonome.

#### IV.1.2.2 Prise en charge des complexités linguistiques

Toutefois certaines difficultés ne sont pas résolues dans cette méthodologie experte : hormis le fait de relever systématiquement le vocabulaire métier, peu d'éléments sont disponibles sur les moyens de transformer du texte en modèle sémantique. Nous avons relevé certaines difficultés linguistiques qu'il faut résoudre au cours de ce processus. Au niveau lexical, les termes métier utilisés dans une règle peuvent être complexes et plusieurs termes peuvent avoir le même sens. Au niveau syntaxique la structure logique de la phrase peut être difficile à dégager, en particulier quand les phrases sont longues. Les anaphores doivent être résolues pour remplacer des pronoms, des adverbes ou certaines tournures nominales par le terme métier auquel ils renvoient. D'autres phénomènes sémantiques interviennent, en particulier l'emploi de termes sous-déterminés pour des raisons stylistiques dont il faut rétablir le sens, ou l'omission de conditions qui sont laissées implicites dans une règle alors qu'elles sont nécessaires à sa formulation exacte. Ces difficultés sont décrite plus longuement dans la section V.3.

### IV.1.3 Un besoin de maintenir les règles

L'état de l'art (voir II.2.3) a montré que la maintenance par les experts nécessite une mise en forme préalable des règles implémentées pour qu'elles puissent être manipulables. C'est la raison pour laquelle ces règles sont d'abord retraduites en langage contrôlé avant d'être confiées aux experts métier. Dans une configuration où des textes sont utilisés comme sources d'information au processus d'acquisition, la retraduction des règles est due uniquement au fait que les textes ont été oubliés une fois règles extraites. Si les textes sont conservés jusqu'à la phase de maintenance, il est possible de leur faire jouer un rôle dans le diagnostic des problèmes. Les règles qui seront alors manipulées par les experts sont les projections textuelles des règles formelles dans les textes réglementaires d'origine. Et la tâche des experts, qui sont familiers avec ces textes d'origine, en serait facilitée.

En outre, comme le travail d'acquisition des règles à partir de textes est un processus long, des erreurs peuvent être commises à plusieurs niveaux. Chacun de ces niveaux peut être à l'origine de problèmes de maintenance si les erreurs ne sont pas corrigées par les acteurs concernés (experts et ingénieurs) avant la fin du développement. Il peut y avoir des erreurs au niveau :

- des textes en langage naturel. Lorsque les textes sources sont incorrects, toutes les transformations qui s'en suivent sur le chemin vers le formel peuvent comporter des erreurs ;
- du passage de la langue naturelle au langage contrôlé. La transformations des règles entre ces deux langages peut produire des erreurs, en particulier si les experts ne maîtrisent pas très bien le langage contrôlé utilisé ou si l'expressivité du langage contrôlé n'est pas adaptée pour traduire la langue naturelle ;
- du passage du langage contrôlé au langage formel. Cette traduction effectuée par les ingénieurs peut aussi produire des erreurs. Dans ce cas les règles formelles résultantes ne sont pas conformes à celles en langage contrôlé ou en langage naturel précédentes.

Ce sont ces erreurs qui produisent des problèmes de maintenance dans une base de règles. Il s'agit notamment de problèmes d'incompréhension, d'incomplétude, d'incohérence et d'évolution. Ils nécessitent le retour aux textes et au processus d'acquisition pour effectuer leur diagnostic et mettre à jour les règles affectées. Nous reviendrons plus en détails sur ces problèmes ainsi que leur diagnostic respectif dans le chapitre V.4.

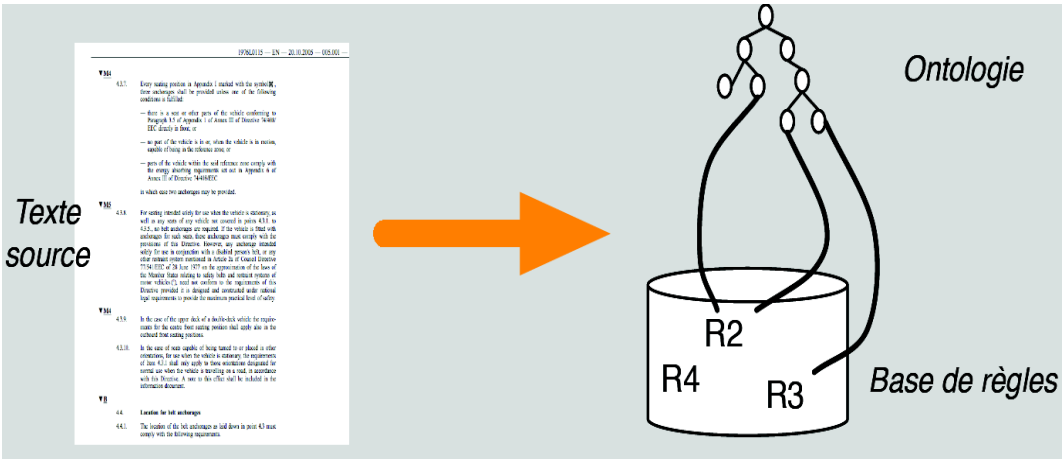



**Figure IV.2** – Un objectif d'extraire à partir des textes.

## IV.2 Notre vision des tâches

Chacune des tâches que nous envisageons, identification, formalisation, maintenance, a déjà été étudiée pour le S.I. (cf. II). La traduction du texte en langage contrôlé fait rarement partie des processus de création et de maintenance des règles tels qu'ils sont étudiés – l'inclusion de cette tâche amont dans le projet ONTORULE a justement été une de ses innovations. Notre conception de la tâche de maintenance s'en trouve élargie : l'explication, l'erreur ou l'évolution peuvent trouver leur source dans le texte, l'erreur peut aussi être due au processus de traduction.

Nous visons une plateforme regroupant la gestion de l'acquisition et de la maintenance de bout en bout ainsi que l'ensemble des sous-tâches qu'elles impliquent. Autrement dit, une plateforme qui prend en entrée un texte réglementaire et qui assiste les experts métier dans la production des règles métier et leur maintenance. Le but est d'avoir une plateforme que nous pourrons mettre à la disposition d'un expert métier afin de l'aider au mieux dans ce travail.

Cette plateforme doit rassembler les techniques et outils nécessaires et adaptés pour réaliser nos tâches. Sa caractéristique principale est son interactivité : tout est centré autour de l'expert pour le guider dans chacune des phases des processus d'acquisition et de maintenance des règles. Un objectif fondamental de la conception de la plateforme est alors de permettre cette interactivité. Deux choses sont essentielles pour cela.

Tout d'abord, les tâches ne sont pas vues comme monolithiques, mais comme une série modulable d'étapes, de traitements interactifs élémentaires. Pour cela, des éléments de méthodologie permettent de cerner l'essentiel des sous-tâches qui constituent une de nos tâches (chapitre V). La tâche ne peut pas être composée à partir de ses étapes sans prendre parti sur ce que sont ces étapes, comment il est possible de les enchaîner. En même temps, nous

ne souhaitons pas contraindre leur ordre plus que nécessaire, ni même limiter la liste exacte des étapes : ces points peuvent dépendre de la nature des textes considérés autant que de la plateforme elle-même. Nous nous contentons d'un *cadre méthodologique* dans lequel les étapes s'insèrent avec souplesse.

Second point, l'interaction ne peut être efficace que si l'expert n'est pas noyé sous les informations. La présentation du texte, de l'ontologie, des règles obtenues, des résultats produits par les outils, est vue comme une navigation dans cet ensemble de données. Cette navigation est rendue possible par l'inventaire des liens entre éléments autres que la proximité dans le texte, et par des outils de recherche ou de sélection des éléments satisfaisant un critère de pertinence donné.

L'ensemble des tâches, aussi bien celles de présentation que les traitements élémentaires, est mis en place sur une structure de données abstraite qui leur est commune et qui permet la modularisation (chapitre VI).

### IV.2.1 Cadre méthodologique d'identification

L'identification consiste en un processus interactif de repérage et de validation des fragments de textes qui pourraient être des règles métier (candidates). L'état de l'art (voir II.2.2.1), nous montre que ces fragments correspondent à des phrases. Le cadre méthodologique est donc un processus itératif de classification des phrases en trois catégories : candidates règles, non candidates, indécidées. Chaque étape est composée d'un filtrage automatique des phrases indécidées dont l'objet peut être de produire des candidates règles ou des phrases non candidates, puis d'une interaction avec l'expert pour confirmer tout ou partie des classements proposés. L'une de ces deux sous étapes peut être vide, soit parce que le filtrage est choisi suffisamment sévère pour ne donner que des résultats certains (quitte à laisser beaucoup d'indécidés), soit à l'inverse parce qu'il n'apporte plus rien pour les phrases indécidées qui restent. Initialement, tout est indécidé. Enfin, chaque étape peut être conçue indépendamment et leur ordre est libre, puisque filtrage et interaction s'appliquent quel que soit l'avancement de la classification ; des exemples de filtrage sont proposés en V.2.

### IV.2.2 Cadre méthodologique de normalisation des règles

La normalisation consiste à traduire les candidates règles en langage contrôlé. On passe donc d'une forme linguistique à une autre moins ambiguë, moins sujette à erreur. Chaque phrase traitée est modifiée et éventuellement divisée en plusieurs phrases. Le cadre méthodologique

qui nous permet d'articuler plusieurs étapes est un processus de reformulation progressive comportant deux types d'opérations :

- les révisions transforment une phrase en une autre plus proche de l'objectif en préservant la sémantique initiale. Une candidate qui n'est traitée que par des révisions produit une seule règle ;
- les divisions produisent plusieurs sous-règles dont chacune véhicule une partie de la sémantique initiale, celle-ci n'étant préservée que si l'on considère l'ensemble des sous-règles obtenues.

Une étape est composée en principe d'un traitement automatique et d'une interaction avec l'expert. Le traitement peut proposer une opération qui reste à valider, par exemple le remplacement d'un nom par l'appellation préférée du concept qu'il représente. Il peut aussi se limiter à sélectionner des candidats pertinents pour une opération qui n'est pas automatisée. Par exemple les phrases formées d'une seule proposition comportant deux compléments d'objet coordonnés pourraient faire l'objet d'une division, mais proposer automatiquement la reformulation est plus difficile que de détecter ses conditions de possibilité.

Nous proposons dans le chapitre V section V.3 un ensemble d'étapes qui peuvent intervenir dans la normalisation. Le cadre méthodologique décrit ici est neutre par rapport à l'ordre des opérations, mais le lecteur sera sans doute convaincu en lisant leur liste que l'ordre est important, même s'il n'y a pas de façon évidente un meilleur choix.

Les différentes étapes produisent un arbre de transformations dont la racine est la phrase candidate initiale et les feuilles sont les règles exprimées en langage contrôlé. Les nœuds ne peuvent cependant pas être considérés isolément par les traitements, dont certains font appel au contexte. Un exemple banal est la résolution d'anaphores, souvent inter-phrastiques. Chaque nœud, bien que ne faisant pas partie du document initial, reste donc relié au contexte de sa racine.

### IV.2.3 Cadre méthodologique de la maintenance

L'idée de maintenance des règles qui est connue dans la littérature (voir II.2.3) repose sur une mise à jour des règles une fois qu'elles sont formalisées ou automatisées. Cette maintenance met surtout à jour les règles lorsque le système dans lequel elles ont été mises en œuvre fonctionne de façon insatisfaisante ou les besoins de l'entreprise évoluent. C'est une tâche généralement décrite en trois étapes : une étape de détection des problèmes, une étape de diagnostic des problèmes et une étape de résolution des problèmes.

Notre idée de maintenance est similaire à cette dernière mais notre maintenance se situe en

complément à la tâche d'acquisition. Il s'agit de pouvoir détecter des problèmes plus tôt dans le processus, de façon à réduire les coûts de maintenance. La détection ne peut donc pas reposer directement sur la découverte d'incohérences formelles ou d'erreurs à l'exécution et leurs traces. Cette détection est vue comme une opération externe qui peut reposer sur des mécanismes de niveaux différents, plus ou moins complets. Nous avons noté plusieurs cas de figure :

- une demande d'explication d'une réponse insatisfaisante, lorsque cette demande est faite au niveau du système, renvoie les règles formelles qui justifient la réponse. Il faut retrouver les phrases candidates qu'elles formalisent ;
- une erreur à l'exécution du système de règles est tracée et renvoie des règles formelles incohérentes. Il faut tracer aussi le processus de traduction des phrases candidates pour voir s'il est fautif ;
- on dispose de patrons syntaxiques d'erreur pour les règles formelles. Certains patrons peuvent être reconnus sur les phrases en langage contrôlé, exactement ou avec une approximation suffisante pour aider l'expert à éliminer ces erreurs avant la traduction formelle ;
- le texte à la source des règles peut aussi être modifié pour suivre l'évolution du domaine. C'est la comparaison de l'ancienne et de la nouvelle version qui permet d'organiser le travail de mise à jour, en ne reprenant la traduction que pour les nouvelles phrases et celles qui ont été modifiées.

La tâche de maintenance est, on le voit, beaucoup plus diversifiée que les autres. Elle exploite essentiellement les dépendances entre versions successives d'une règle candidate, dont les principes de représentation sont décrits dans la section IV.2.4, et la recherche de configurations sur ces représentations.

### IV.2.4 Le rôle des annotations

Nous visons un processus d'acquisition (voir IV.2) des règles qui prend en entrée en plus des textes une ontologie qui décrit le vocabulaire métier de domaine. La combinaison de l'ontologie et des textes, comme l'indexation conceptuelle dans la recherche d'information, favorise la compréhension et l'exploration des textes. Cette combinaison repose sur une annotation des textes au regard de l'ontologie et sert de structure de base à l'acquisition qui l'enrichit au fur et à mesure que les règles sont extraites. Nous sommes partis d'une idée d'annotation "textes-ontologie" pour améliorer l'exploration des textes et au final nous produisons une structure d'index qui est utilisée pour une gestion interactive de nos tâches.

### Une ontologie pour décrire le vocabulaire métier

La construction de l'ontologie précède celle des règles. En effet, l'ontologie représente le vocabulaire métier et sert à stabiliser la sémantique terminologique du texte. Les ontologies que nous utilisons sont construites suivant la méthode Terminae de construction d'ontologie à partir de textes. C'est un travail parallèle au nôtre, dont nous exploitons les résultats [4].

La méthode Terminae repose sur la liste des termes les plus pertinents du domaine métier du texte qui sont extraits à partir d'outils de traitement automatique de la langue (TAL). Ensuite, ces termes sont normalisés pour donner naissance à un réseau de termes dits termino-concepts via des relations qui décrivent des liens syntaxiques ou sémantiques entre termino-concepts ou des propriétés d'un termino-concept. La normalisation de la liste initiale des termes consiste, à partir d'outils d'analyse syntaxique et terminologique (TreeTagger, Yatea), à déterminer les termes métier, les entités nommées (termes métier de type nom de personne, de lieu, ou d'objet, etc) et les relations terminologiques. Une classification est effectuée sur l'ensemble des termes où les termes métier et les entités nommées sont regroupés en termino-concepts et les relations terminologiques en groupes de relations termino-conceptuelles. Le termino-concept regroupe des termes synonymes et un terme préférentiel représente le groupe. Une fois tous les termino-concepts obtenus, ils sont organisés sous forme de réseau suivant des liens de subsomption (hiérarchiques) entre termino-concepts et des relations associatives. Il n'y a pas à ce niveau de distinction entre concept et instance. Une relation termino-conceptuelle peut être construite de la même manière qu'un termino-concept avec des groupes de relations terminologiques synonymes. Enfin, la phase de formalisation consiste à transformer les termino-concepts en concepts et en instances, et les relations termino-conceptuelles en relations conceptuelles. Au final, nous est fourni une ontologie formelle, dite lexicalisée, étendue par des informations terminologiques qui caractérisent les apparitions de ses éléments (concepts, instances, relations) dans les textes réglementaires[Ma et al., 2009, Ma et al., 2010].

### Un support construit autour d'une structure d'index

Une fois l'ontologie construite, elle est utilisée pour stabiliser la sémantique du texte. Cela passe par l'annotation sémantique du texte au regard de l'ontologie. Comme celle-ci est associée à un ensemble d'informations terminologiques pour chacun des éléments de l'ontologie, l'annotation sémantique permet de connecter chaque terme du texte à une entité de l'ontologie suivant les informations fournies par l'ontologie étendue. Toutefois comparé à l'état de l'art (voir III.3)

4. Remerciements à Nouha Omrane, ma collègue dont la thèse porte sur la construction de l'ontologie à partir de textes réglementaires.

où l'annotation d'un texte au regard d'une ontologie se résume le plus souvent à l'identification des entités nommées du texte et leur association à des instances de concept, nous avons une annotation plus étendue que la classique. La figure IV.3 en montre un exemple. Dans une optique de compréhension et d'exploration des textes, et surtout de traduction des règles d'origine, tous les termes métier sont importants. Et comme dans l'ontologie ces termes peuvent correspondre à la fois à des concepts, à des relations conceptuelles ou à des instances de concepts, il faut projeter toutes les catégories d'éléments ontologiques sur un texte réglementaire.

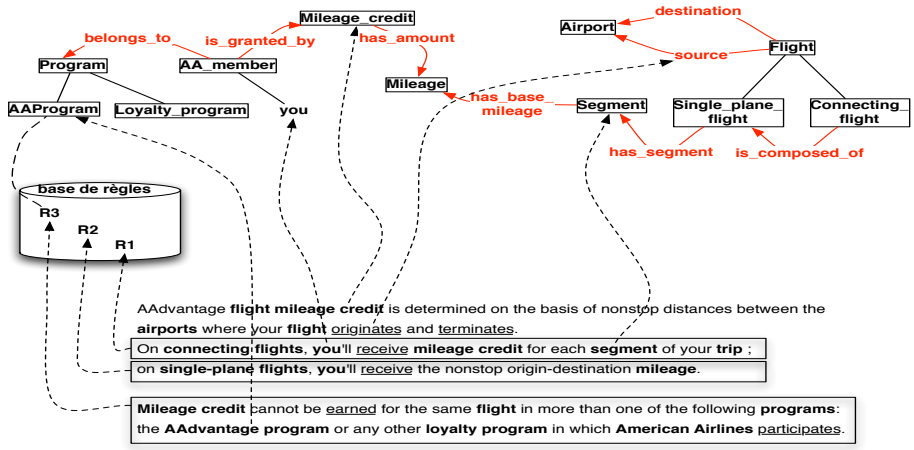



**Figure IV.3** – Exemple d'annotation d'un fragment textuel au regard d'une ontologie et des règles.

Une telle annotation définit une structure d'index qui interconnecte les textes à l'ontologie. Mais cette structure se trouve étendue par des annotations supplémentaires tout au long du processus d'acquisition des règles ([Guissé et al., 2009], [Lévy et al., 2010]). En effet, comme les règles sont extraites des textes annotés, elles sont toutes aussi initialement annotées par les mêmes éléments conceptuels que ceux annotant les fragments textuels d'où elles dérivent. Ensuite au fur et à mesure qu'elles sont normalisées, elles peuvent être à nouveau annotées par d'autres éléments conceptuels. D'où l'enrichissement de la structure d'index par ces annotations interconnectant règles et ontologie. D'un autre côté, les règles, du simple fait qu'elles dérivent des textes et qu'elles sont appelées à être modifiées par des opérations de normalisation qui les simplifient et les précisent, sont aussi des connaissances sémantiques qui peuvent être connectées aux textes. Cela favorise un meilleur suivi du processus d'acquisition.

Au final, nous avons une structure triangulaire d'index (voir IV.4) avec des textes connectés à une ontologie ainsi qu'à toutes les règles dérivées et avec des règles annotées au regard de l'ontologie. Nous nous appuyons sur cette structure pour organiser toutes les opérations de manipulation des données (textes, règles et ontologie) dans les tâches d'acquisition et de maintenance, en assurant l'interactivité avec l'expert.

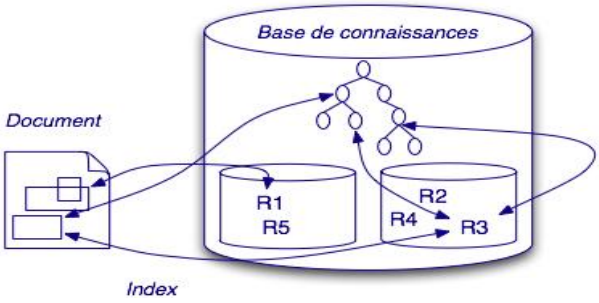



**Figure IV.4** – Structure d'index pour texte réglementaire.

## IV.3 Conclusion

Ce chapitre a permis de faire le point sur notre problématique à partir d'une lecture des travaux qui se relient à l'acquisition et à la maintenance des règles à partir de textes. Nous avons présenté notre vision des tâches d'acquisition et de maintenance, ainsi que l'organisation d'une réponse reposant sur une méthodologie interactive pour chaque tâche et une structure de données permettant de garantir une interaction avec l'expert en même temps qu'une souplesse de mise en œuvre et une intégration simplifiée des outils nécessaires à la réalisation des tâches. Nous reviendrons plus en détails sur tous ces aspects de mise en œuvre de notre plateforme d'aide dans les chapitres suivants.

# Chapitre V

# Méthodologies d'acquisition et de maintenance des règles métier

## V.1 Cycle de vie d'une règle candidate

Le cycle de vie des règles dans le développement d'une application (voir II.2.2) montre un processus d'acquisition des règles de façon progressive. Notre travail se focalise sur une configuration où les règles sont décrites dans des textes réglementaires. Le but est de les identifier à partir de ces textes réglementaires et d'aider à les formaliser afin de faciliter leur automatisation en langage machine. Autrement dit, l'on se retrouve dans un processus par lequel les règles sont transformées d'une représentation textuelle en langage naturel vers une représentation en langage contrôlé, puis une représentation formelle facilement automatisable. Ce passage n'est pas direct, le processus de transformation est long et difficile mais il repose sur un schéma cohérent, progressif et faisant intervenir à chacune de ses étapes les acteurs ayant l'expertise nécessaire pour réaliser cette étape.

Concrètement, ce processus d'acquisition se décrit en trois étapes. Il débute par une première étape d'exploration des textes pour en identifier les fragments textuels pertinents dont certains sont validés par l'expert métier comme règle métier candidate. Cette étape est suivie d'une seconde de normalisation où les règles candidates sont traduites dans un langage contrôlé par l'expert qui tente de simplifier la langue naturelle dans laquelle les règles identifiées sont rédigées dans le but de les rendre lisibles et facilement compréhensibles pour les acteurs de l'étape suivante. Ces derniers sont les ingénieurs qui interviennent à la troisième étape. Ils sont chargés de formaliser les règles dans un langage formel à partir duquel commence le travail d'implémentation. Cependant, notre travail ne couvre pas cette dernière étape mais se concentre

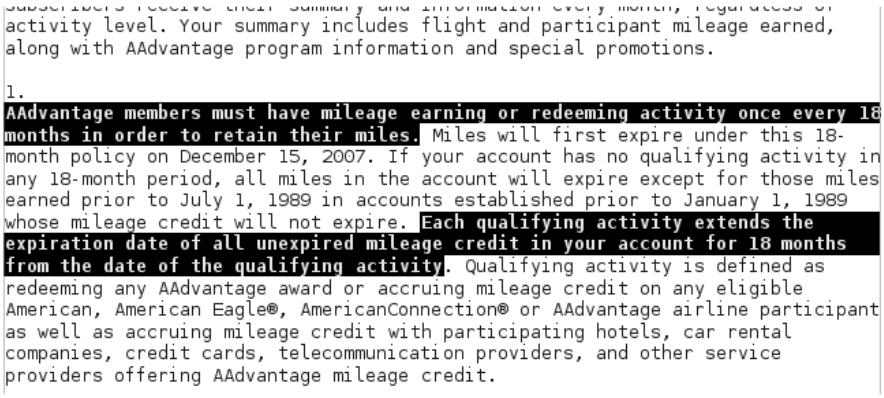
sur l'identification des règles et leur transformation en langage contrôlé.

Les règles extraites et transformées doivent demeurer cohérentes, pertinentes et conformes aux besoins définis dans les textes initiaux. Pour cela, nous proposons de conserver l'historique des transformations de normalisation pour vérifier la qualité des règles par rapport aux textes d'origine. Cet historique simplifie la mise à jour des règles pour corriger les problèmes d'incomplétude, d'incohérence et d'évolution que celles-ci peuvent comporter.

### V.1.1 Règles dans le texte

Au début du processus d'acquisition, les règles métier sont rédigées en langage naturel et contenues dans des textes réglementaires (voir II.1.3). A ce stade, les règles ne sont pas structurées et elles existent dans des formats libres. Ce sont des fragments textuels, en général des phrases d'un texte réglementaire. Les experts les produisent par rapport à leur connaissance du domaine, indépendamment du SGRM, et ne se situent pas nécessairement dans une visée applicative déterminée.

La figure V.1 montre quelques exemples de règles métier figurant dans un extrait du corpus AAdvantage, et qui ont été identifiées (marquage par un fond noir). Dans les textes les règles métier expriment des contraintes et conditions sur « ce que l'entreprise sait d'elle-même » (Business Rules Manifesto) ou plus clairement sur le vocabulaire métier du domaine.


activity level. Your summary includes flight and participant mileage earned, along with AAdvantage program information and special promotions.

1.
**AAdvantage members must have mileage earning or redeeming activity once every 18 months in order to retain their miles.** Miles will first expire under this 18-month policy on December 15, 2007. If your account has no qualifying activity in any 18-month period, all miles in the account will expire except for those miles earned prior to July 1, 1989 in accounts established prior to January 1, 1989 whose mileage credit will not expire. **Each qualifying activity extends the expiration date of all unexpired mileage credit in your account for 18 months from the date of the qualifying activity**. Qualifying activity is defined as redeeming any AAdvantage award or accruing mileage credit on any eligible American, American Eagle®, AmericanConnection® or AAdvantage airline participant as well as accruing mileage credit with participating hotels, car rental companies, credit cards, telecommunication providers, and other service providers offering AAdvantage mileage credit.

**Figure V.1** – Fragment de texte du corpus AAdvantage.

### V.1.2 Règles candidates initiales

Une "règle candidate" est une règle métier non encore formalisée (donc sous une forme textuelle libre ou contrôlée) mais qui a déjà été identifiée dans les textes réglementaires et validée par l'expert en tant que règle. Cette appellation est conforme aux travaux cités précédemment (voir II.2.2.1) sur la détection des règles ; elle étend cependant l'emploi du terme en incluant les versions partiellement transformées (voir la section V.1.3). Les règles sont identifiées suivant une méthode de fouille axée sur une exploration syntaxique et sémantique des textes. Les fragments textuels pertinents de règles qui sont détectés ainsi sont filtrés manuellement par l'expert qui sélectionne les règles candidates initiales.

Concrètement, l'identification (voir V.2) des règles candidates s'effectue dans un environnement d'aide à l'expert pour la détection de tous les passages réglementaires. L'expert a la possibilité de sélectionner directement les passages qu'il juge valides mais nous lui proposons aussi diverses techniques de fouille que nous classons en deux catégories. D'une part, nous avons une fouille textuelle à base de patrons linguistiques construits via des mots-clés caractéristiques de règles (qui peuvent être tirés des langages contrôlés comme SBVR). Ces patrons s'appliquent sur les arbres syntaxiques des phrases des textes ou sur sur leur contenu brut, et seules les phrases présentant ces caractéristiques sont alors filtrées. D'autre part, nous avons une fouille sémantique à base des concepts du domaine qui consiste à filtrer les phrases où apparaissent un concept donné. Au final, le travail de l'expert aboutit à la validation des phrases qui constituent nos règles candidates initiales.

### V.1.3 Reformulation des règles candidates

Pour poursuivre dans le processus d'acquisition, les règles candidates initialement identifiées dans les textes sont fournies en entrée à l'étape de normalisation en langage contrôlé qui précède la formalisation des règles. Pour cela :

– Nous visons une normalisation qui correspond en un ensemble de spécifications progressives traduisant les règles candidates initiales dans un langage contrôlé dans le but de réduire les complexités de la langue naturelle dans laquelle les règles candidates initiales sont transcrites. La normalisation permet de désambiguïser et de simplifier chacune des règles candidates de façon à ce qu'elle soit autonome, élémentaire et facilement compréhensible, et ce indépendamment des textes d'origine. Pour cela, la normalisation agit à plusieurs niveaux de la structure linguistique de la règle candidate notamment aux niveaux lexical, syntaxique et sémantique. Et chacune des règles candidates peut faire

l'objet de plusieurs opérations de normalisation suivant les types de complexités qu'elle contienne.

– Nous ne choisissons pas un langage formel spécifique pour formaliser les règles, parce que à notre connaissance il n'y a pas de noyau commun à tous les systèmes d'inférence des SGRM qui ait un pouvoir expressif suffisant. Dans ONTORULE, les partenaires formalisaient le même problème différemment selon les capacités d'inférence de leur langage formel, en tenant compte des besoins de raisonnement de l'application. Nous nous limitons à une normalisation poussée qui consiste à reformuler chacune des règles candidates simplifiées de façon à faciliter sa traduction formelle.

Le travail de normalisation repose sur un langage cible qui nous permette d'atteindre nos objectifs. Notre choix s'est inspiré de SBVR. La règle reformulée n'utilise que des termes métier du domaine, des faits non ambigus portant sur ces termes et les connecteurs logiques de SBVR et ses modalités, éventuellement des variables utilisées comme des entités nommées. Ces formulations sont le plus souvent directement traduisibles dans la plupart des langages formels de règles (voir II.2.2.2).

### V.1.4 Contrôle de qualité

Une fois la règle candidate normalisée elle est directement soumise à la phase de formalisation. Mais la règle est aussi maintenue à partir de sa version normalisée en langage contrôlé. La maintenance d'une règle permet de garder sa conformité avec les besoins du système d'information décrits dans les textes d'origines. Dans certains cas, c'est la maintenance de la version normalisée qui entraîne celle de la version formelle. Comme, en règle générale, le processus de maintenance consiste à identifier des problèmes dans une base de règles (ensemble de règles), à les diagnostiquer et à les corriger, nous avons essentiellement considéré des problèmes (voir IV.1.3) d'incohérences et d'inconsistances dont le diagnostic permet de rétablir la qualité des règles extraites. Nous proposons de diagnostiquer certains problèmes déjà au niveau des textes. Concrètement, nous cherchons à aider les experts métier dans cette tâche. Ce sont eux qui analysent les problèmes, les identifient et mettent à jour les règles affectées. Nous les aidons à effectuer le diagnostic de ces problèmes à partir de la trace des multiples reformulations que les règles peuvent subir tout au long du processus de normalisation. Pour ce faire, le travail d'acquisition des règles aboutit au stockage de l'historique de toutes les versions produites par reformulation afin de pouvoir revenir sur les traces de chacune des règles.

## V.2 Identification des règles candidates

Pour identifier les règles candidates contenues dans les textes réglementaires, notre objectif consiste à assister les experts dans ce travail de fouille. Il est possible d'imaginer que les experts fassent le travail à la main en sélectionnant tous les passages réglementaires des textes. Mais cela n'est possible que sur des textes de petite taille. Sur des textes volumineux, la détection manuelle est plus difficile car elle peut prendre du temps. C'est la raison pour laquelle nous mettons à leur disposition des techniques de fouille (voir IV.1.1) pour améliorer le travail. Ces techniques permettent de filtrer les phrases pertinentes présentant des caractéristiques de règles métier, et elles sont de deux types : les techniques de fouille syntaxique et sémantique. La première fouille repose sur l'analyse de la structure syntaxique des phrases et la seconde sur l'analyse de la sémantique lexicale des phrases. L'analyse d'une phrase s'effectue en appliquant un patron linguistique sur son contenu brut (la chaîne de caractères), ou sur son contenu syntaxique après une analyse syntaxique, ou sur son contenu lexical de domaine (les termes métier qu'elle contient). Un seul patron est appliqué à la fois dans une base documentaire pour en filtrer les phrases concernées.

Les techniques de fouille (syntaxique et sémantique) que nous utilisons ici sont pour la plupart déjà connues dans la littérature. Notre objectif est surtout d'expérimenter celles qui peuvent servir à identifier les règles candidates et à aider l'expert dans son travail.

Une fois filtrées, la phrases pertinentes sont soumises à une validation métier par l'expert. Celui-ci valide les phrases qui sont effectivement des règles métier, et qui devront dans l'étape suivante être réécrites. Les phrases validées par l'expert sont alors les versions initiales des règles candidates. Ces phrases ainsi que celles jugées invalides sont retirées de la base documentaire avant l'application d'un autre patron.

### V.2.1 Définition des patrons linguistiques

Les patrons linguistiques sont construits autour d'éléments linguistiques caractéristiques de règles métier. Un premier critère est la présence des mots-clés caractéristiques de règles que nous pouvons tirer des langages contrôlés comme SBVR (voir III.1.2) ou RuleSpeak, ou la présence de termes métier de domaine particulièrement significatifs de règles. Le repérage des mots-clés se généralise à une fouille syntaxique. Dans ce cas, les patrons sont définis sur un contenu textuel éventuellement enrichi par une information syntaxique. Le patron qui s'applique sur le contenu brut d'une phrase n'est pas le même que celui qui s'applique sur le résultat de l'analyse syntaxique de la phrase. Les termes métier sont utilisés quant à eux pour effectuer une fouille

sémantique.

#### V.2.1.1 Fouille syntaxique

Une fouille syntaxique correspond à une fouille à base de mots-clés caractéristiques de règles métier. Chaque mot-clé permet de construire un patron spécifique afin de retrouver toutes les phrases contenant ce mot-clé. Un mot-clé est manipulé dans un patron selon que celui-ci s'applique sur le contenu brut des phrases ou sur leurs arbres syntaxiques respectifs. L'application d'un patron suppose alors une analyse lexicale ou une analyse syntaxique de chacune des phrases de la base documentaire (textes réglementaires).

**L'analyse lexicale** permet d'utiliser une phrase comme une suite de mots ou de considérer la chaîne de caractères associée. Dans ce cas, un mot-clé est un mot simple ("must") ou un groupe de mots successifs ("shall be") ou non successifs ("if...then"). Les deux premiers types de mots-clés sont détectés par une simple recherche de chaîne dans la phrase. Les mots non successifs peuvent être traités comme une conjonction (***and*** : sélectionner toutes les phrases qui contiennent tous les mots de la conjonction à la fois), ou comme une disjonction (***or*** : sélectionner toutes les phrases qui contiennent au moins un mot de la disjonction).

On peut définir un peu plus précisément ce type de patron.

Considérons les ensembles **M** et **S** qui définissent respectivement l'ensemble des mots-clés caractéristiques de règles et l'ensemble des phrases du document. Pour nos expériences, nous avons choisi l'union des mots clés de SBVR-SE et des mots clés de RuleSpeak.

$M$ est découpé en trois sous-ensembles correspondant chacun à un type de mots-clés : $M = \{T_1 \cup T_2 \cup T_3\}$ où $T_1$ est la liste des mots-clés simples (un seul mot chacun), $T_2$ la liste des mots-clés constitués de groupes de mots successifs, $T_3$ la liste des mots-clés constitués de mots ou groupes de mots disjoints.

L'ensemble **P** des patrons construits avec les mots-clés contenus dans **M** est

$$\mathbf{P} = \{p | p = OP_{i=1}^{n} m_i \; et \; m_i \in T_1 \cup T_2\} \text{ où } \mathbf{OP} \text{ est l'un des } \{+, \cdots, \vee, \wedge\}$$

Nous utilisons aussi la fonction $norm(s)$ qui est une normalisation de la phrase $s$ obtenue en remplaçant tous les séparateurs par le caractère blanc (d'espacement).

L'application d'un patron de **P** pour identifier les phrases concernées peut alors se définir

comme une fonction $r$ de recherche :

$$\begin{aligned} r : P &\rightarrow S \\ p \rightarrow r(p) &= \{s \in S / match(s,p) = 1\} \end{aligned}$$

où

$$match(s,p) = \begin{cases} 1 \quad \text{si} \begin{cases} OP = + & \text{et } p \text{ est une sous chaîne de } norm(s) \\ OP = \cdots & \text{et pour tout i, } m_i \text{ est une sous chaîne de } norm(s), \\ & \text{et ces sous chaînes apparaissent dans l'ordre} \\ OP = \wedge & \text{et pour chaque i } m_i \text{ est une sous chaîne de } norm(s) \\ OP = \vee & \text{et pour au moins un i } m_i \in s \end{cases} \\ 0 \quad \text{sinon} \end{cases} \tag{V.1}$$

**L'analyse syntaxique** permet d'utiliser l'arbre syntaxique d'une phrase et d'y appliquer des patrons plus complexes que ceux utilisés pour les contenus bruts des phrases. Elle permet d'effectuer une fouille sur la base des informations syntaxiques associées à une phrase notamment les catégories morpho-syntaxiques des mots ainsi que les dépendances entre ces mots. Les patrons utilisés dans ce cas sont construits à l'aide de mots-clés caractéristiques de règles et de descripteurs syntaxiques. Ils s'appliquent sur l'arbre syntaxique de la phrase.

Il existe deux types d'arbres syntaxiques (voir Figure V.2 [1]) :

1. les arbres syntaxiques de composants qui associent une étiquette morpho-syntaxique (S : Sentence (phrase), NP : Noun Phrase (groupe nominal), VP : Verb Phrase (groupe verbal), N : Noun (nom), V : Verb (verbe) , etc) à des sous parties d'une phrase ;
2. et les arbres de dépendance qui spécifient les dépendances syntaxiques ou les types de relations syntaxiques entre les mots et, indirectement, entre les différents composants d'une phrase (sujet(,), COD(,), apposition(,), etc).

Des exemples de patrons sont :

– **NP + can + only + V $\cdots$ if + NP**. Ce patron s'applique sur un arbre syntaxique des composants et il permet de chercher toutes les phrases contenant les mots-clés "can", "only"et "if" avec la précision que "can" vient après un groupe nominal, est

1. Exemple tiré de http ://en.wikipedia.org/wiki/Parse_tree et modifié

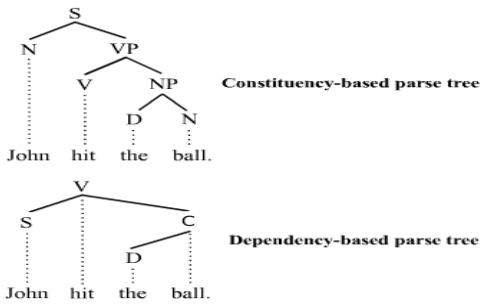



**Figure V.2** – Exemples arbres de composants et arbres de dépendances.

immédiatement suivi de "only" puis du verbe, après lequel se trouve, en position non nécessairement contiguë, "if" suivi d'un autre groupe nominal.

– **sujet(S,V) $\wedge$ S + shall + V** . Celui-ci s'applique sur un arbre de dépendance et il permet de retrouver toutes les phrases où le mot-clé "shall" s'intercale exactement entre le dépendant et la tête d'une dépendance 'sujet' – c'est-à-dire entre le sujet et le verbe.

De la même manière que au niveau de l'analyse lexicale, nous pouvons définir les patrons spécifiques aux arbres syntaxiques :

Considérons les mêmes ensembles **M** de mots clés et **S** de phrases utilisés précédemment. Considérons de plus les ensembles $\mathbf{S}_{asc}$ et $\mathbf{S}_{asd}$ qui contiennent respectivement les arbres syntaxiques de composants et les arbres de dépendance résultant de l'analyse syntaxique des phrases de **S**.

L'ensemble **P** des patrons comprend deux sous-ensembles $\mathbf{P}_{asc}$ et $\mathbf{P}_{asd}$. Ils sont construits avec les mots-clés contenus dans **M** et les composants de l'un des deux types d'arbres syntaxiques. Chaque patron filtre une suite ou une somme de combinaisons de mots-clés et de composants, ou de mots-clés et de dépendances syntaxiques.

$\mathbf{P}_{asc} = \{p|p = OP_{i=1}^{n}(m_i|c_i) \wedge_{j=1}^{k} ancestor(c_i, c_j)$ et $m_i \in M$ et $c_i \in \{NP, VP, N, V, etc\}$ et $OP \in \{+, \cdots\}$
$\mathbf{P}_{asd} = \{p|p = OP_{i=1}^{n}(m_i|c_i) \wedge_{k=1}^{l} rel_k(c_i, c_j)$ et $m_i \in M$ et $c_i \in \{S, V, C, etc\}$ et $OP \in \{+, \cdots\}$ *et* $rel_k \in \{Sujet, Object, Attribut, etc\}\}$

Un patron comporte donc une partie qualifiante (avant le $\wedge$) et une partie relationnelle. L'application d'un patron de **P** pour identifier les phrases concernées peut alors se définir

comme une fonction $r$ de recherche :

$$\begin{aligned} r : P &\rightarrow S \\ p \rightarrow r(p) &= \{s \in S / match(s,p) = 1\} \end{aligned}$$

où

$$match(s,p) = \begin{cases} 1 \quad \text{si} \begin{cases} \exists\ s \in S & s \text{ vérifie la partie qualifiante du patron } p \\ & \text{et } s_{asc} \text{ sa partie relationnelle} \\ \exists\ s \in S & s \text{ vérifie la partie qualifiante du patron } p \\ & \text{et } s_{asd} \text{ sa partie relationnelle} \end{cases} \\ 0 \quad \text{sinon} \end{cases} \tag{V.2}$$

#### V.2.1.2 Fouille sémantique

La fouille sémantique dont nous parlons ici n'a pas encore été expérimentée dans les travaux de détection de règles métier mais elle apparaît dans les travaux sur la recherche d'information axés sur l'indexation conceptuelle (voir III.3). Elle repose sur la sémantique lexicale de la phrase. Elle consiste à effectuer des recherches non pas avec des mots-clés caractéristiques de règles métier mais avec des termes métier du vocabulaire de domaine. Notre but est de compléter la fouille syntaxique par une fouille à partir des termes : ce qui permettra de trouver certaines phrases pertinentes non identifiables par mots-clés.

Contrairement aux mots-clés, il n'y a pas de configuration connue *a priori* de termes métier qui soit caractéristique de règles. Les patrons définis dans la fouille syntaxique peuvent être utilisés pour apprendre certaines configurations. Cependant, comme les phrases sélectionnées par la fouille syntaxique ne sont que des candidates plausibles à être une règle métier, effectuer une fouille sémantique avec les termes obtenus ainsi risque de donner un bruit important dans les résultats. Il nous a paru plus pertinent de réduire l'effet de ce bruit importé dans les données initiales en considérant les termes métier figurant dans des règles reconnues.

Nous proposons de chercher d'abord les termes métier qui caractérisent le mieux les règles candidates déjà extraites. C'est une fois que les premières fouilles syntaxiques à base de mots-clés ont été validées que peut commencer une fouille sémantique. Ainsi des calculs sémantiques sont à effectuer dans la base de règles candidates afin d'identifier les termes métier dont les

fréquences dans les règles et dans le reste du texte diffèrent le plus, à défaut simplement les plus fréquents, et ensuite ces termes pourront être utilisés dans le cadre d'une fouille sémantique pour recherche les phrases qui contiennent ces termes.

Il existe une autre façon aussi intéressante de faire une fouille sémantique. Elle consiste à comparer individuellement les phrases restant dans le textes (c.-à-d.. celles qui ne sont pas apparues dans les fouilles) et les règles candidates extraites suivant les termes métier qu'elles partagent. Il peut être utile d'analyser les phrases qui partagent les mêmes termes que ceux d'une règle candidate donnée ou qui ont plusieurs termes en commun avec une règle candidate.

### V.2.2 Rapport avec l'utilisateur

Nous avions déjà précisé que l'identification des règles candidates est une tâche effectuée par l'expert métier. Il guide le processus et valide ou non les phrases résultant de ces fouilles en règle candidate. Nous n'envisageons pas un processus automatique, parce qu'aucune technique ne donne des résultats assez fiables, et l'intervention humaine est toujours nécessaire pour valider (voir IV.1.1), Le processus est semi-automatique et interactif. L'expert contrôle toujours le travail de fouille, y compris l'arrêt et le volume de travail effectué. Il est chargé de choisir le type de fouille (syntaxique ou sémantique) le plus adéquat ou le plus à son goût, de sélectionner les patrons qu'il a envie d'appliquer, et de valider ensuite les résultats, puis de relancer ou pas un cycle de fouille. L'application des patrons quant à elle est à la charge de la plateforme (voir VI.2). Le système est aussi chargé après chaque fouille, après chaque application d'un patron, après validation des phrases détectées ou leur non validation, de réduire l'espace de recherche et ainsi d'améliorer les fouilles futures. Cette réduction se résume à une simple différence ensembliste : $S = S - (S_{inv} \cup S_v)$ où $S_{inv}$ correspond à l'ensemble des phrases jugées invalides par l'expert et $S_v$ à l'ensemble des phrases jugées valides à partir desquelles des règles candidates ont été dérivées.

## V.3 Vers la formalisation des règles candidates

Une fois les règles candidates extraites des textes, elles subissent un certain nombre de transformations ou de reformulations afin qu'elles soient faciles à formaliser dans un langage de règles. Ces reformulations d'une règle candidate initiale comportent trois grandes catégories d'opérations : une première qui correspond à une normalisation de la règle candidate dans un langage contrôlé, pour nous SBVR-SE ; une deuxième qui décompose la règle candidate

si nécessaire en plusieurs autres règles candidates élémentaires ; et enfin une troisième qui reformule la règle candidate pour obtenir une structure textuelle plus proche du formel.

## V.3.1 Normalisation

La normalisation consiste à traduire les règles candidates initiales en SBVR-SE dans le but de rendre le langage naturel suffisamment désambiguïsé et simplifié et surtout facilement compréhensible par les ingénieurs dans la phase de formalisation. La normalisation s'effectue tant au niveau lexical que syntaxique.

Nous avons avons identifié quatre types de transformations à effectuer pour normaliser les règles candidates. Une normalisation lexicale remplace les termes d'une règle candidate par des termes plus authentiques du vocabulaire métier de domaine. Une normalisation contextuelle consiste à restaurer dans la phrase les références contextuelles extérieures. Elle permet d'expliciter une règle candidate en levant tous les implicites et imprécisions qu'elle peut contenir, qui la rendent incompréhensible en dehors des textes d'origine. Une normalisation syntaxique permet de simplifier la structure syntaxique de la règle candidate. Une normalisation sémantique permet de clarifier toutes les entités du discours qui ne sont pas explicitées dans les textes et qui nécessitent d'être prises en compte pour rendre la règle candidate la plus autonome possible.

Les exemples utilisés ici sont tirés des corpus AAdvantage et Audi.

### V.3.1.1 Normalisation lexicale

La normalisation lexicale consiste à stabiliser le vocabulaire métier utilisé dans les règles candidates initiales afin qu'il soit conforme au vocabulaire métier de domaine. C'est un travail de réduction du vocabulaire, souvent de désambiguîsation et de précision sur les termes utilisés dans la règle candidate. Elle s'applique quand le texte n'utilise qu'un fragment d'une appellation plus complexe, ou quand il utilise plusieurs dénominations alternatives ayant le même sens. Dans le premier cas la solution est de corriger le terme concerné en le remplaçant par le terme complet et correct du vocabulaire de domaine. S'il s'agit de dénominations multiples, il faut remplacer le terme ambigu ou imprécis par le terme synonyme le plus adéquat sémantiquement.

Prenons le cas suivant, nous avons une règle candidate initiale (1) qui subit une normalisation lexicale pour donner une règle candidate reformulée (2) plus précise :

> (**1**). *Two* ***belts*** *or* ***restraint systems*** *are required for the* ***buckle inspection*** *and the* ***low-temperature buckle test****.*

Les termes métier "buckle inspection" et "low-temperature buckle test" de la règle existent déjà dans le vocabulaire de domaine et donc il n'y a pas lieu de les remplacer. Par contre, les termes "belts" et "restraint systems" n'existent pas dans le vocabulaire de domaine, ils sont ambigus et leurs valeurs sémantiques (dans cette phrase) sont représentées dans ce vocabulaire. Ils sont alors remplacés par des termes métier précis qui sont respectivement "seat belts" et "child restraint systems" :

> (**2**). *Two* ***seat belts*** *or* ***child restraint systems*** *are required for the* ***buckle inspection*** *and the* ***low-temperature buckle test****.*

### V.3.1.2 Normalisation contextuelle

La normalisation contextuelle est une extension de la normalisation lexicale. Si cette dernière stabilise les termes métier utilisés dans une règle candidate, la normalisation contextuelle quant à elle agit sur le lexique implicite de la règle candidate notamment les mots et les symboles utilisés en langage naturel et renvoyant à des termes métier présents ailleurs dans la phrase ou dans d'autres phrases. Ces mots (ou symboles), appelés très souvent contextuels, rendent implicites des informations sans lesquelles toute règle candidate devient incompréhensible. Or le travail de formalisation extrait la règle candidate du texte et donc la présente loin de son contexte, rendant le vocabulaire contextuel très difficile à comprendre. La normalisation contextuelle sert alors à rétablir ce contexte qui manque à la règle candidate. De cette façon, la règle candidate se suffit à elle-même et peut alors être comprise indépendamment des textes.

Nous avons identifié cinq types de références susceptibles d'apparaître dans les règles candidates et pour lesquelles une normalisation contextuelle est nécessaire.

1. **Mots grammaticaux**. Ces mots correspondent à des pronoms (sujet ou complément) qui renvoient à des termes métier actifs du domaine en particulier les entités. Ce sont des cas d'anaphores utilisés en langage naturel pour éviter la répétition ou la redondance de certains termes dans les textes. Ces mots peuvent être des pronoms personnels sujet (**I**, **You**, **It**, **They**, etc), des pronoms personnels complément (**Me**, **You**, **It**, **Them**, etc), des pronoms relatifs (**Whom**, **Which**, **That**, **This**, etc), etc.

   La normalisation contextuelle consiste alors en une résolution des anaphores à partir du contexte de la règle candidate dans son texte d'origine. Dans l'exemple suivant une règle candidate est dérivée de la seconde phrase.

   > *All the adjustment devices shall undergo a strength test as prescribed in paragraph 7.5.1.* ***They*** *must not break or become detached under the tension set up*

*by the prescribed load.*

Le mot "They" est un pronom et l'analyse de son contexte qui est ici la phrase précédente montre qu'il renvoie au terme métier "All the adjustment devices". La normalisation de la seconde phrase est

> *The adjustement device must not break or become detached under the tension set up by the prescribed load.*

A ce stade, la règle n'est pas encore autonome. Le dernier type de références reprend cet exemple.

2. **Clés de référence**. Elles sont des symboles ou des numéros qui renvoient à d'autres fragments textuels. Dans une règle candidate (pour ce qui nous intéresse ici), elles renvoient à des endroits du texte d'origine sans lesquels la règle candidate serait incomplète et incomprise. La normalisation contextuelle consiste alors à prendre en considération les références sur lesquelles pointent ces clés pour compléter la règle candidate et enlever ces clés de la règle. Considérons l'exemple suivant :

> (7.5.1) *The buckle shall be connected to the tensile-testing apparatus and the load shall then build up to 980 daN ... If the buckle is part of the attachment,* ***the buckle shall be tested with the attachment****, in conformity with paragraph 7.5.2. below,*
>
> (7.5.2) *The attachments shall be tested in the manner indicated in paragraph 7.5.1., but the load shall be 1,470 daN.*

On peut voir que dans la règle candidate (1), la clé "7.5.2" renvoie au paragraphe dans le texte ayant ce numéro (expression soulignée). La règle candidate définit les conditions de charge ("load") pour tester la boucle ("buckle"). Puis elle précise que si la boucle est solidaire de la fixation d'une ceinture ("attachment", il faut prendre en compte le paragraphe 7.5.2. Et celui-ci renvoie au paragraphe précédent mais fixe une valeur de charge différente. En l'espèce, mais ce n'est pas toujours le cas, la clé de référence introduit une exception. Une normalisation contextuelle de la règle candidate donne :

> *If the buckle is not part of the attachment, il shall be connected to the tensile-testing apparatus and the load shall then be build up to 980 daN*
>
> *If the buckle is part of the attachment, it shall be connected to the tensile-testing apparatus and the load shall then build up to 1,470 daN.*

3. **Relations cachées**. Il arrive fréquemment que des termes métier qui figurent dans la même règle candidate soient liés par des relations, et que ses relations ne soient pas explicitées. La normalisation contextuelle permet alors de remettre les termes manquant dans la règle candidate ainsi que les liens entre les termes existant déjà dans la règle.

   Dans l'exemple suivant, entre la règle candidate (1) et sa version reformulée (2) par une normalisation contextuelle, nous pouvons voir qu'un "Mileage credit" est gagné pour un vol bien spécifique, d'où le lien "awarded for" entre "Mileage credit" et "flight". En plus, ce vol est pris par un "AAdvantage member" qui bénéficie du "Mileage credit", d'où le lien "takes" entre "AAdvantage member" et "flight".

   (**1**). ***Mileage credit*** *will be credited only to the account of the AAdvantage member who flies, etc.*

   (**2**). *Mileage credit* ***awarded for a flight*** *will be credited only to the account of the AAdvantage member who* ***takes that flight****, etc.*

4. **Termes métier «génériques»**. Ce sont des termes métier qui lorsqu'ils apparaissent dans une règle candidate renvoient à des concepts très généraux du vocabulaire de domaine et cela donne un sens et des contraintes trop vagues à la règle candidate qui est finalement ambiguë. Le but de la normalisation contextuelle est de considérer le contexte de la règle candidate dans le texte et vérifier s'il n'existe pas de termes plus spécifiques dans ce contexte. Si tel est le cas, les termes généraux sont remplacés par les termes spécifiques. Dans l'exemple suivant, la règle candidate (1) se définit autour du terme métier "Test" qui est général d'autant que dans le vocabulaire de domaine il existe divers types de tests. Le contexte, qui est ici la phrase précédente dans le texte, montre que c'est un test bien spécifique dont il question ici : le "micro-slip test". D'où la version reformulée de la règle candidate en (2).

   *The samples to be submitted to the micro-slip test shall be kept for a minimum of 24 hours in an atmosphere having a temperature of 20 ±5 °C and a relative humidity of 65 ±5%. (**1**)* ***The test*** *shall be carried out at a temperature between 15 and 30 °C*

   *(**2**)****The micro-slip test*** *shall be carried out at a temperature between 15 and 30 °C*

5. **Relations de discours** Il arrive aussi que le contexte doive être restauré sans qu'il y ait de marqueur visible. Le lien au contexte est le fait d'une relation de discours, une

affirmation poursuivant la description commencée dans la précédente - mais constituant une règle distincte. C'est le cas dans l'exemple que nous avions laissé inachevé :

> *The adjustement device must not break or become detached under the tension set up by the prescribed load.*

En l'état, on ne peut vérifier si la règle est satisfaite ou pas, car on ne sait pas de quel test il s'agit ! Dans la forme initiale :

> *All the adjustment devices shall undergo a strength test as prescribed in paragraph 7.5.1.* ***They*** *must not break or become detached under the tension set up by the prescribed load.*

la seconde phrase est une élaboration de la première. Elle précise ce que signifie *subir un test de résistance comme prescrit au paragraphe 7.5.1.* Une normalisation plus complète est donc :

> *During the strength test, the adjustement device must not break or become detached under the tension set up by the prescribed load.*

Il reste à expliciter la relation cachée de *prescribed load* à *strength test*.

#### V.3.1.3 Normalisation Syntaxique

Après la normalisation des mots dans une règle candidate vient la normalisation syntaxique qui consiste à simplifier la structure syntaxique de la règle. Elle s'inspire des travaux de simplification syntaxique de texte (voir IV.1.2.2). Dans ces derniers, plusieurs opérations ont été proposées pour simplifier la structure d'ensemble d'une phrase. Elles couvrent la plupart des complexités du langage naturel liées à la structure mais quelques unes de ces complexités apparaissent très souvent dans les textes réglementaires.

La normalisation syntaxique repose sur trois opérations de simplification : la structuration d'une règle candidate si les éléments linguistiques qui la composent sont désordonnés, le découpage de la règle candidate en sous règles semblables, et le découpage de la règle candidate en plusieurs autres règles candidates élémentaires.

1. **Structuration**. Elle permet de rétablir un ordre canonique des idées et consiste à réordonner les différentes propositions de la règle candidate, en général en plaçant la proposition jugée la plus importante en première position et en la faisant suivre de toutes les propositions secondaires. Ce réarrangement s'effectue en général en ajoutant de nouveaux connecteurs (tels que des pronoms relatifs, des conjonctions de subordination, des virgules

de juxtaposition, etc) entre les propositions, qui viennent s'ajouter à ceux qui existaient déjà.

L'exemple suivant montre une règle candidate (1) où "Upgrades" est manquant dans la proposition qui exprime les conditions, puis sa version structurée (2) où l'ellipse est en même temps résolue. :

(**1**).*Upgrades* ***are void if*** *sold for cash or other consideration.*

(**2**).***If*** *upgrades are sold for cash or other consideration, these upgrades* ***are void****.*

2. **Découpage des énumérations**. Les énumérations sont fréquentes dans les textes réglementaires. Elles résument dans la même phrase ce qui pourrait être plusieurs propositions indépendantes. Il existe divers types d'énumérations construites par des connecteurs tels que *either...or*, *neither...nor*, *not only...but also*, *whether...or*, *and*, *or*, etc., des ponctuations de juxtaposition (les virgules), etc.

   Le découpage des énumérations consiste à diviser la règle candidate en autant de sous règles candidates qu'il y a de termes dans l'énumération. Chacune des sous règle candidate est alors définie par une proposition. Le but de la normalisation syntaxique ici est de décomposer une règle candidate en plusieurs autres règles candidates élémentaires et indépendantes. On parle de règle candidate élémentaire lorsque celle-ci ne peut plus à son tour être décomposée.

   Dans l'exemple suivant, la règle candidate (1) se subdivise en trois règles candidates élémentaires (2).

   (1). ***Neither*** *accrued mileage,* ***nor*** *award tickets,* ***nor*** *upgrades are transferable by the member upon death.*

   (2). *Accrued mileage is not transferable by the member upon death. Award tickets are not transferable by the member upon death. Upgrades are not transferable by the member upon death.*

3. **Découpage d'interdépendances**. Il ne s'agit plus d'énumérations, mais de découper une règle candidate trop longue. En effet, en simplification textuelle une phrase longue pose toujours des problèmes de compréhension car elle englobe plusieurs informations ou idées à la fois. Le but est alors de subdiviser la règle candidate, jusqu'ici constituée d'une seule phrase, en plusieurs petites phrases. Une fois le découpage effectué, la règle candidate correspond à l'ensemble des petites phrases.

   Un cas typique concerne des règles candidates longues qui décrivent une multitude de propositions interconnectées les unes aux autres par des pronoms relatifs ((*Who*, *Which*,

etc)) ou juxtaposées. L'exemple suivant d'une règle candidate (1) reformulée en (2) le montre très bien :

(**1**). *The membership year, **which** is the period in **which** your elite benefits are available, runs from March 1 through the last day of February of the following year.*

(**2**). *The **membership year** is a period. **Member**'s elite benefits are available in the **membership year**. The membership year runs from March 1 through the last day of February of the following year.*

#### V.3.1.4 Normalisation sémantique

La normalisation sémantique vient apporter des précisions à une règle candidate à la suite de formulations issues des autres types de normalisation. Le plus souvent cela arrive dans le cas d'une normalisation syntaxique notamment quand il s'agit de découper une énumération. Dans ce cas, il arrive souvent qu'une notion soit partagée par les sous règles candidates élémentaires résultant du découpage alors qu'elle n'est pas repérée comme terme métier et que donc le vocabulaire pour expliciter la coréférence manque. Le but de la normalisation sémantique est alors est de créer un terme métier nouveau comme support du partage de la notion par les sous règles candidates. La définition du terme métier peut produire une règle candidate supplémentaire, en général une règle structurelle qui décrit ses propriétés.

Dans l'exemple suivant, la règle candidate (1) est reformulée en (2) puis en (3). La première reformulation est le découpage d'une énumération mais les deux règles candidates résultantes partagent une entité décrite par le même terme ("two perpendicular axes"). L'explicitation du démonstratif ("These") nécessite de lui donner une identité. D'où une normalisation sémantique dans laquelle, nous créons un terme pivot ("Sensitivity test axes") via une nouvelle règle candidate élémentaire ("Sensitivity test axes are perpendicular").

(**1**). *When retractors are being tested for sensitivity to vehicle deceleration they shall be tested at the above extraction along two perpendicular axes, **which** are horizontal if the retractor is installed in a vehicle as specified by the safety-belt manufacturer.*

(**2**). *When retractors are being tested for sensitivity to vehicle deceleration they shall be tested at the above extraction along two perpendicular axes. **These two perpendicular axes** are horizontal if the retractor is installed in a vehicle as specified by the safety-belt manufacturer.*

(**3**). *When retractors are being tested for sensitivity to vehicle deceleration they shall be tested at the above extraction along the* ***sensitivity test axes****.* ***Sensitivity test axes*** *are perpendicular.* ***Sensitivity test axes*** *are horizontal if the retractor is installed in a vehicle as specified by the safety-belt. manufacturer.*

### V.3.2 Formulation logique en If-Then

A la suite des opérations de normalisation, une règle candidate est suffisamment désambiguïsée, simplifiée et précisée pour être autonome, élémentaire et facilement compréhensible. Pour la rapprocher de la structure des règles formelles, nous imposons de reformuler à nouveau la règle candidate pour séparer prémisses et conclusions. Cette dernière reformulation rendrait plus simple la formulation de la règle candidate en SBVR-SE et sa traduction vers les langages formels de règles.

Pour cela, comme nous l'avions décrit dans III.1.4 et V.1.3, nous proposons de reformuler la règle candidate sous la forme décrit comme suit :

1. La traduire sous une forme **If** - **Then** dans laquelle on identifiera facilement et clairement la prémisse et la conclusion.
2. La traduire en une formulation logique qui consiste à éliminer de la règle candidate autant que possible les éléments linguistiques autre que les termes métier de domaine, les connecteurs logiques (and, or, not, etc), les quantifieurs (each, exactly, etc), les opérateurs modaux (It is necessary that, etc), le **If** et le **Then**. Ce uniquement si la sémantique est totalement préservée.

Considérons la règle candidate (1) suivante obtenue après normalisation en SBVR-SE. Dans la formulation (2), nous l'avons d'abord mise sous une forme If-Then avant de ne garder que les mots nécessaires à la compréhension de la règle et enlever les mots en noir sauf le mot "these" qui risque de recréer une ambiguïté déjà levée dans la phase de formalisation.

(**1**). ***When upgrades are sold*** *for* ***cash****, these* ***upgrades are void****.*
(**2**). ***If upgrades are sold*** *for* ***cash Then*** *these* ***upgrades are void****.*
(**2**). ***If upgrades have been sold Then*** *these* ***upgrades are void****.*

### V.3.3 Rapport utilisateur

Le travail de normalisation est à effectuer par un expert métier, c'est à lui de rendre les règles manipulables pour les ingénieurs qui les formalisent. Les opérations de normalisation lui

permettent de rendre les règles candidates compréhensibles au niveau lexical (normalisation lexicale), autonomes (normalisation contextuelle), syntaxiquement simplifiées et élémentaires (normalisation syntaxique), sémantiquement cohérentes (normalisation sémantique). Analyser le travail étape par étape permet une plus grande systématisation, un meilleur contrôle de la complétude du travail, et réduit l'expertise logique impliquée dans cette première phase. Il est difficile d'imposer un ordre dans l'application des opérations mais il est souvent recommandé dans les travaux sur la simplification textuelle de simplifier les mots avant de simplifier la phrase en entier.

Actuellement, l'expert est en mesure d'effectuer ces opérations à la main. Ces normalisations peuvent s'opérer spontanément et sans distinguer de types de normalisation, c'est souvent ainsi que les experts travaillent, en ajustant la forme finale par essais et erreurs. Mais notre objectif est, pour des documents complexes, de proposer à l'expert une assistance . Un système interactif lui propose sur chaque type de normalisation, une détection ou une transformation automatique qu'il pourra ensuite accepter, modifier si nécessaire et valider, ou alors refuser.

## V.4 Maintenance des règles

### V.4.1 Recours aux textes

Nous savons que d'une part, la méthode d'acquisition (présentée dans V.3) de règles à partir des textes réglementaires présente les règles sur plusieurs niveaux. Autrement dit, chaque règle est soumise à plusieurs révisions à travers le processus de reformulation, de sa version initiale dans les textes à sa version complètement normalisée. C'est cette dernière version qui est prise en compte dans la phase de formalisation. Ce processus peut être à l'origine de plusieurs erreurs qui renvoient à un besoin de maintenance des règles. Il s'agit d'erreurs qui apparaissent dans les textes ou dans le passage du langage naturel au langage contrôlé à travers les opérations de normalisation.

Et d'autre part, un schéma de maintenance (voir IV.1.3) consiste en général à maintenir des règles déjà mises en œuvre, c'est-à-dire des règles formalisées et automatisées. Le processus de maintenance quant à lui consiste en trois phases, une phase de détection des problèmes présents dans la base de règles formelles, une phase de diagnostic de ces problèmes et une phase de mise à jour des règles.

Ainsi, nous nous intéressons particulièrement ici à la prise en charge des problèmes liés aux erreurs produites dans notre processus d'acquisition. Notre travail ne porte ni sur la détec-

tion des problèmes ni sur leur correction mais nous proposons une méthode de diagnostic des problèmes au regard des textes d'origine et de l'ensemble des reformulations que subissent les règles candidates initiales.

Cette méthode repose sur la traçabilité que nous utilisons pour diagnostiquer les problèmes d'incohérences dans la base de règles. Elle consiste à recourir aux traces d'une règle candidate donnée en revenant sur l'ensemble de ses versions de normalisation antérieures jusqu'à sa version initiale dans les textes. Cela permet de remonter chacun des problèmes afin de comprendre son origine et de pouvoir le justifier. Les problèmes d'incohérences sont dues à des erreurs commises à certaines étapes du processus d'acquisition. Une partie de ces erreurs correspond à des incorrections dans les textes d'origine, et l'autre partie apparaît quant à elle dans les étapes de normalisation lorsque des problèmes d'interprétation et d'explicitation conduisent à une non conformité entre les textes et les règles dans leur version transformée. Si ces erreurs persistent jusqu'à la fin du processus, elles se traduisent pendant l'utilisation du SGRM par des problèmes de maintenance.

La traçabilité peut aussi être utilisée lorsque des règles deviennent obsolètes et ne servent plus à rien avec le temps. Ces règles posent alors problèmes et il nécessaire de les mettre à jour en faisant recours aux textes pour analyser s'il faut les modifier ou les supprimer de la base de règles. Cela est du au fait que les textes changent régulièrement et donc toutes les règles dérivées des fragments ayant subi des modifications devront alors être mises à jour.

### V.4.2 Diagnostic des problèmes

Nous prenons en compte uniquement les problèmes susceptibles d'apparaître dans la base de règle et dont le diagnostic peut être effectué en recourant aux textes d'origine. Nous avons ainsi sélectionné les types de problèmes où la traçabilité peut aider à comprendre s'ils proviennent des textes ou du processus de reformulation. Ces problèmes sont bien connus dans la littérature et nous pouvons les classer en trois catégories :

- les problèmes d'incompréhension et d'incomplétude qui caractérisent des règles qui quand elles sont appliquées par exemple dans un SGRM aboutissent à des décisions difficilement compréhensibles ou incomplètes ;
- les problèmes d'incohérences qui traduisent des contradictions entre des couples de règles de la base, qui lors qu'ils sont appliqués produisent des décisions incorrectes ou alors les règles d'un couple ne peuvent s'appliquer en même temps ;
- les problèmes d'évolution qui conduisent à des règles obsolètes qui nécessitent une mise à jour et qui sont caractérisés par évolution du fonctionnement de l'entreprise.

Nous analysons ici plusieurs exemples de problèmes et pour chacun d'eux nous tenterons de voir comment le diagnostic peut se faire en utilisant la traçabilité. Ces exemples sont tirés pour la plupart des corpus AAdvantage et Audi, les autres sont tirés d'autres textes réglementaires qui ne sont pas décrits dans ce rapport.

#### V.4.2.1 Incohérence

Les règles incohérentes sont des règles dont l'application est soit impossible, soit incertaine, soit sans effet. Il y a nécessairement à l'origine de ce fait une mauvaise rédaction. Au premier rang des incohérences, on trouve les contradictions qui, permettant de déduire une chose et son contraire, vident le système logique de tout résultat. Décider si un ensemble de règles contient des règles contradictoires ne peut se faire que par des méthodes de déduction automatique au niveau des règles formalisées. Typiquement dans le SGRM, la contradiction est détectée à ce niveau et ensuite tracée. Les autres incohérences ont été répertoriées par les praticiens de l'écriture des règles en fonction de leur expérience. Elles ont des formes syntaxiques qui permettent de les reconnaître plus tôt, et c'est pour cela que nous nous sommes intéressés à leur reconnaissance avant même que la formalisation soit achevée.

Nous n'avions pas assez d'incohérences dans nos cas d'usage pour avoir une idée de comment elles se produisent. Nous supposons que là aussi le retour aux textes permet aux experts de corriger les erreurs. Détecter certaines incohérences au point de départ du processus d'acquisition est un gain : la détection n'est pas complète mais elle a lieu avant la formalisation. Lorsque les problèmes sont identifiés dans la base de règles, leur coût de réparation est, de l'avis des experts, beaucoup plus important.

Nous décrivons ci-dessous quelques problèmes d'incohérence classiques :

**Les règles indécidables** décrivent des règles dont la traduction en une procédure de décision est ambiguë. L'exemple suivant pose une difficulté au moment de la décision. Dans la règle (1), l'obtention du statut de résident est obtenu à la suite d'un certain délai ("12 months") de résidence en territoire norvégien. Cette condition est distincte de celle de la règle (2) qui confère le statut à partir du moment où l'immigré s'installe en Norvège, à un moment où on ne sait pas si les conditions d'obtention sont remplies.

> (**1**) **Norwegian resident is defined as one who is staying in Norway, when the stay is intended to last or has surpassed 12 months.**
> (**2**) **A person who moves to Norway is considered a resident from the date of arrival.**

**Les redondances** décrivent des relations d'inclusion entre deux règles c'est-à-dire lorsque dans un couple de règles, le domaine d'application de l'une est couvert par celui de l'autre. En général, il s'agit de couples où une règle est une spécification ou vice versa une généralisation de l'autre. Dans l'exemple suivant la règle (2) est une spécification de la première (1) car les "Emerald customers" sont aussi des "Loyalty customers". Comme on le voit dans cet exemple, ces inclusions sont dues aux relations de subsomption entre les termes métier de domaine.

(**1**) **Loyalty customers can use the frequent customers boarding queue.**
(**2**) **Emerald customers can use the frequent customers boarding queue.**

**Les incompatibilités** décrivent des règles dont chacune isolément serait sans problème, mais telles que, prises ensemble, l'une au moins devient inapplicable. Il est inutile de les laisser subsister simultanément dans la base de règles, et il y a vraisemblablement une erreur dans l'interprétation du texte. Le couple de règles suivant le montre très bien.

(**1**) **A man is not a woman.**
(**2**) **If a human being is a man and a woman, he is an hermaphrodite.**

#### V.4.2.2 Obsolescence

Dans la durée, les besoins du système d'information changent régulièrement, certaines règles métier deviennent obsolètes et nécessitent d'être modifiées. Soit que le texte soit modifié par les responsables, soit que les conditions d'application de la règle ne soit plus réunies et qu'elle devienne sans objet. Dans le texte réglementant les bonus d'une compagnie aérienne, on trouve :

> **If your account has no qualifying activity in any 18-month period, all miles in the account will expire except for those miles earned prior to July 1, 1989 in accounts established prior to January 1, 1989 whose mileage credit will not expire.**

Ce texte s'analyse en deux règles, applicables aux bonus obtenus avant et après le 1er juillet 1989. 23 ans après cette date, il est vraisemblable que la première sera bientôt obsolète et qu'il va devenir inutile de mentionner l'exception. Pour maintenir la cohérence, il est nécessaire de vérifier l'existence de règles textuelles qui mentionnent l'expiration des bonus dans leur conclusion (ce qui pourrait rendre la modification totalement ou partiellement sans effet) ou leur prémisse (si l'obsolescence est confirmée, cette règle sera elle aussi obsolète).

Quand le texte est modifié, il s'agit de répercuter à moindre coût ce changement sur les éventuelles règles qui en sont dérivées. Le texte sur les ceintures de sécurité a été modifié à 13

reprises pendant les 4 dernières années. Les modifications classiques changent une formulation, modifient la valeur d'un paramètre, précisent une condition délicate ou ajoutent un cas particulier (l'Airbag a été introduit plusieurs années après la ceinture). Le processus de mise à jour comporte trois étapes :

- suppression des révisions et sous-règles de la phrase modifiée ;
- construction d'un nouvel ensemble de révisions et sous-règles ;
- vérification de la cohérence de la nouvelle version des règles avec le reste du texte.

### V.4.3 Interaction avec l'utilisateur

Nous utilisons une configuration classique de gestion de la maintenance des règles où on identifie les problèmes, effectue leur diagnostic et met à jour les règles affectées. Notre but est d'assister l'expert métier dans cette tâche. Toutefois, c'est à l'expert d'identifier les problèmes, soit *a priori* dans les textes d'origines soit *a posteriori* dans la base de règles, et de corriger ces problèmes. Nous l'assistons dans le diagnostic des problèmes en mettant à sa disposition l'historique des transformations afin qu'il puisse analyser les problèmes et leur origine, et ensuite mettre à jour les règles à partir du point où les problèmes ont été détectés. L'identification de ce type de problèmes permet de compléter les règles incomplètes, de garder les règles les plus pertinentes et cohérentes pour le système d'information, et de modifier ou supprimer celles qui le sont moins (règles incohérentes) ou celles qui deviennent obsolètes avec le temps. L'historique des reformulations quant à elle est stockée tout au long du processus d'acquisition des règles et elle est soumise à l'expert à sa propre demande. L'historique correspond à l'ensemble des versions de règles qui précédent une règle normalisée jusqu'au fragment textuel d'où elles dérivent ainsi que le contexte [2] de ce dernier fragment dans les textes d'origines.

## V.5 Conclusion

Nous avons présenté dans ce chapitre une méthodologie d'acquisition des règles métier à partir de ressources textuelles réglementaires. Cette méthodologie couvre essentiellement l'identification des règles dans les textes et leur prise en charge en vue de leur formalisation future. Sur ce dernier point, le but est de formuler les règles de façon à ce qu'elles soient facilement compréhensibles et traduisibles dans un langage formel.

Nous proposons cette méthodologie pour assister les experts chargés de faire ce travail.

2. Le contexte correspond ici à l'ensemble des phrases qui entourent une règle donnée dans un texte

- D'une part, elle facilite l'identification des règles métier en s'appuyant sur des techniques de fouille textuelle. Il s'agit notamment de techniques de fouille syntaxique à base de patrons linguistiques et de fouille sémantique à base des termes métier du vocabulaire de domaine.
- D'autre part, elle permet de mettre les règles sous une forme désambiguïsée, précise, autonome, élémentaire et facilement manipulable par les ingénieurs chargés de les formaliser. Pour cela, la méthodologie propose l'utilisation d'un langage contrôlé. En outre, elle inclut un processus de normalisation qui consiste à réduire les complexités de la langue naturelle et qui guide l'écriture des règles en SBVR-SE sans que les experts soient obligés de connaître ce langage contrôlé. Enfin, la méthodologie propose une extension de la normalisation qui consiste à traduire les règles dans des formulations logiques en If-Then. Cela rapproche la forme des règles normalisées de celle des langages formels de règles.

Nous complétons cette méthodologie par un contrôle de qualité des règles extraites. En effet, le processus d'acquisition étant long et difficile, de nombreux problèmes de maintenance liés à la complétude des règles, à leur cohérence ainsi qu'à leur évolution peuvent apparaître. La correction de ces problèmes permet de rétablir la qualité des règles. Nous proposons d'assister les experts dans le diagnostic des problèmes pour faciliter la mise à jour des règles. Pour cela, nous conservons toutes les traces de reformulation des règles afin de pouvoir remonter tout type de problème jusqu'aux textes et analyser s'il provient de ces derniers ou durant la normalisation. Les experts pourront ainsi maintenir toutes les règles impactées par un problème donné.

Nous parlons de méthodologie parce qu'en tentant d'assister les experts nous développons et testons les éléments adéquats à l'acquisition des règles à partir desquels :

- Nous identifions les problèmes liés à la prise en charge des textes notamment les complexités du langage naturel.
- Nous expérimentons des méthodes qui ont déjà été utilisées dans la littérature et qui ne sont pas toujours généralisables aux textes que nous rencontrons.
- Nous tentons de réunir tous les outils nécessaires dans chacune des étapes de l'acquisition et qui permettent d'améliorer le travail des experts à travers un système semi-automatique.

# Chapitre VI

# Conception et développement de la plateforme SemEx

Dans ce chapitre, nous présentons la conception et le développement de la plateforme SemEx qui met en œuvre les méthodologies exposées au chapitre précédent.

Pour la conception, il s'agit de décrire d'abord le modèle de données sur lequel nous nous appuyons pour organiser les opérations qui réalisent les tâches d'acquisition et de maintenance, puis la modularisation de ces tâches permettant de décomposer les opérations en modules et de structurer leurs enchaînements et leurs dépendances. Le support de données est une structure d'index généralisé interconnectant les textes d'origine aux ressources sémantiques telles que l'ontologie et les règles candidates extraites. Et pour couvrir les deux tâches, nous avons un module d'identification des règles, un module de normalisation de ces règles, un module de maintenance et un module d'accès qui définissent les fonctionnalités de manipulation des données.

Pour le développement, il s'agit de décrire la représentation formelle de la structure d'index et l'implémentation des modules de traitements. Ce dernier point consiste à implémenter les méthodologies qui régissent le fonctionnement des modules et à intégrer tous les outils nécessaires à créer un cadre interactif d'aide à l'exploration des textes réglementaires, à l'identification des règles à partir de ces textes, à la normalisation de ces règles et à leur maintenance quand les textes évoluent ou lorsque les règles ne sont pas en conformité avec les textes.

## VI.1 Structure d'index

Nous proposons le terme structure d'index (voir IV.2.4) pour désigner la structure de données sur laquelle reposent les opérations de réalisation des tâches d'acquisition et de maintenance des règles. Elle sert à mettre en correspondance les textes, l'ontologie descriptive du vocabulaire métier et les règles candidates dérivées des textes ([Lévy et al., 2010]). La structure d'index supporte tout le processus d'édition des règles, de l'exploration des textes pour en extraire les règles, en passant par les opérations de normalisation, jusqu'à leur mise à jour ([Lévy et al., 2010, Nazarenko et al., 2011]). Elle définit un ensemble de liens interconnectant ces deux modèle et son exploitation permet de :

- Faciliter l'exploration et la compréhension sémantique des textes et des règles en considérant leurs liens avec l'ontologie.
- Tracer les normalisations que peuvent subir une règle de sa version d'origine dans les textes jusqu'à une forme en langage contrôlé complètement stabilisée (précise, autonome et élémentaire). Cela permet en particulier le suivi des règles une fois acquises et facilite la gestion de leur maintenance notamment pour faire le diagnostic des problèmes d'incohérence depuis les textes.
- Naviguer entre les différentes ressources, ce qui favorise une exploration sémantique plus approfondie de ces dernière à travers les liens qu'elles partagent. En effet, l'index définit un espace de liens permettant de retrouver toute phrase source d'une règle candidate donnée et vice versa. Il permet de lister les concepts ontologiques qui apparaissent dans une règle ou de lister les règles dans lesquelles un concept donné apparaît. De la même manière il est possible de naviguer entre les éléments ontologiques (concept, instance ou rôle) et leurs termes associés dans le texte.
- Documenter les règles extraites afin de pouvoir les justifier et vérifier leur conformité avec les textes d'origine.

Pour préciser la structure d'index, il est nécessaire de décider comment s'effectue cette correspondance. En premier lieu, quelles catégories d'unités textuelles peuvent faire l'objet d'une annotation et vice versa quels éléments conceptuels peuvent annoter une unité textuelle. Il faut aussi spécifier les éléments structurels du document qui seront pris en compte, ainsi que la nature des liens de correspondance possibles ou, autrement dit, les informations qui seront portées par le lien lui-même. Pour cela, nous avons spécifié trois modèles (Figure VI.1) : un modèle de document, un modèle sémantique constitué d'une ontologie et d'une base de règles, et un modèle de correspondance qui les lie ([Lévy et al., 2010]).

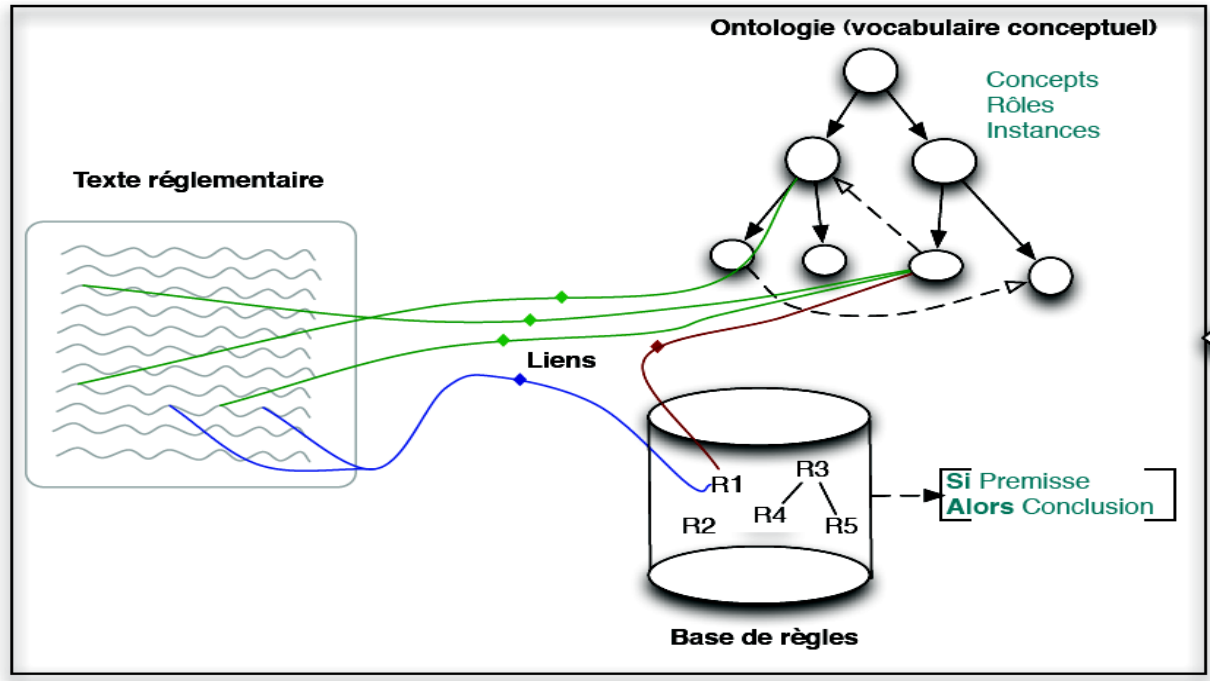



**Figure VI.1** – La structure d'index.

### VI.1.1 Modèle de document

Le document est pour nous un texte réglementaire. L'idée est d'avoir une structure de document qui facilite la manipulation des fragments textuels pertinents pour l'acquisition des règles métier. Celles-ci sont généralement décrites sous forme de phrases dans les textes mais les mots associés aux termes du vocabulaire métier défini dans l'ontologie ainsi que les mots associés aux mots-clés caractéristiques de règles dans les langage contrôlé, sont tous d'une grande importance dans la réalisation de cette tâche.

Le modèle de document détermine quels fragments de texte constituent des unités documentaires qui peuvent être liés à des éléments ontologiques (concepts, instances ou relations) ou à des règles métier. En principe, n'importe quelle liste de caractères peut être annotée mais on se limite en général à des intervalles continus. On peut aussi typer les séquences de caractères en question pour distinguer des mots, des syntagmes, des phrases, des paragraphes, des sections de document, etc. La structure d'index dépend largement du modèle de document considéré.

Nous considérons un modèle de document défini à partir de la structure logique de celui-ci, ce qui permet de représenter le document sous forme d'arbre. Un document est alors divisé par exemple en sections, chaque section en sous sections, ainsi de suite jusqu'à l'unité élémentaire qui est la phrase. Chaque élément est repéré par un identifiant. Si le lien porte sur un fragment de texte plus réduit que la phrase, par exemple quand il s'agit de mots ou de groupes mots,

et comme il est très encombrant de créer *a priori* un identifiant pour chaque mot, le début de phrase est alors considéré comme une adresse de référence relativement à laquelle nous utilisons des coordonnées de localisation du fragment pour l'identifier.

### VI.1.2 Modèle sémantique

Le modèle sémantique indique quelles unités sémantiques peuvent être associées aux unités documentaires. Il est constitué de deux ressources sémantiques à savoir une ontologie et une base de règles métier. Une unité sémantique peut donc être un élément de l'ontologie pris parmi ses concepts, ses relations et ses individus, ou alors un élément de la base de règles. Cette dernière inclut l'arbre des règles candidates construit par les transformations décrites en V.3. Elle comporte aussi en principe la version formalisée de la règle, mais la traduction dans un langage de règles sort du cadre de notre étude ; les structures de données permettant de l'intégrer à l'index peuvent sans doute être construites en utilisant les variantes de RIF pour représenter les règles.

#### VI.1.2.1 L'ontologie

Il s'agit d'une ontologie de domaine qui sert à décrire le vocabulaire métier utilisé dans les textes et les règles. L'ontologie constitue un modèle formel qui sert de référence aux vocabulaires hétérogènes du domaine métier pouvant apparaître dans les textes et susceptibles d'être utilisés dans la reformulation des règles par des experts métier différents. Autrement dit, l'ontologie permet de mettre en place un vocabulaire métier stable et commun à tous les textes et à toutes les règles, et de spécifier certaines propriétés sémantiques de ce vocabulaire. Dans le monde des règles métier, l'ontologie est aussi très utilisée pour ses atouts de raisonnement logique qui favorisent l'inférence de nouvelles connaissances sémantiques, la mise à jour de la base de règles, le calcul des similarités sémantiques entre les règles, etc, atouts que nous utilisons dans le processus de maintenance des règles candidates (voir VI.7).

Pour des raisons citées plus haut (voir IV.2.4), nous considérons ici des ontologies construites à partir de textes et nous disposons d'une ontologie lexicalisée c'est-à-dire d'une ontologie à laquelle sont associées des informations linguistiques. Ces dernières correspondent à un ensemble de groupes de termes métier synonymes pour exprimer les éléments conceptuels de l'ontologie. Ces termes décrivent les occurrences de ces éléments dans les textes réglementaires.

Nous utilisons la structure classique d'une ontologie (FigureVI.2) qui organise les concepts du domaine métier dans un graphe suivant des relations sémantiques et de subsomption entre ces

concepts. Les informations linguistiques, qui décrivent une terminologie suivant des relations de synonymie entre des termes métier regroupés autour des concepts (respectivement les relations ou les individus), ne sont pas incluses dans la structure de l'ontologie afin de garder la dimension conceptuelle de celle-ci. Elles sont décrites dans un thésaurus dans lequel nous répertorions tous les termes métier associés aux éléments conceptuels.

#### VI.1.2.2 La base de règles

La base de règles désigne l'ensemble des règles candidates dans toutes leurs versions. Nous préservons l'historique de toutes les reformulations pour une meilleure gestion du suivi du processus d'édition des règles depuis les textes. Pour ce faire, la base de règle repose sur une structure de graphe qui permet d'organiser le stockage des règles extraites ainsi que leurs dépendances.

Chaque règle candidate sélectionnée ou identifiée dans les textes d'origine donne naissance à de nouvelles versions de règles candidates chaque fois qu'elle subit une révisions ou une décomposition pour sa normalisation (voir Figure VI.3). A travers des normalisations lexicales, contextuelles, syntaxiques ou sémantiques successives, une règle candidate garde son état textuel d'origine (dans les textes) qui est reformulé plusieurs fois pour la rendre autonome, indépendante des textes et sémantiquement précisée, pour la décomposer en des règles candidates élémentaires qui peuvent elles-mêmes être reformulées jusqu'à mettre en place une version éditée dans une formule logique "**If-Then**". Dans l'exemple VI.4, la règle candidate R13 est révisée trois fois, une première version R14 résultant d'une normalisation lexicale, une seconde R15 résultant d'une normalisation syntaxique de structuration de la précédente, et deux autres R16 et R17 résultant d'une normalisation syntaxique de décomposition en règles candidates élémentaires.

L'ensemble des règles candidates est stocké dans une structure de graphe (Figure VI.5. Voir VI.4.2 pour plus de détails) où sont représentées les dépendances entre les règles candidates liées à l'historique des reformulations, et des propriétés associées à chaque règle candidate. Les règles candidates sont liées entre elles suivant trois relations : la relation "**previousForm**" pour dire qu'une règle candidate est une révision d'une autre règle candidate, la relation "**subRule**" pour dire qu'une règle candidate est obtenue à partir d'une autre règle candidate à la suite d'une décomposition (suivie éventuellement de révisions), et la relation "**OriginalRuleVersion**" qui connecte toute version dérivée d'une règle candidate à sa version initiale. Chaque règle candidate possède en outre les propriétés suivantes : la propriété "**ID**" confère un identifiant unique à la règle candidate, la propriété "**content**" donne le contenu textuel de la règle candidate, la propriété "**type**" indique son type qui peut être 'structurelle', 'opératoire', 'dérivationnelle', etc,

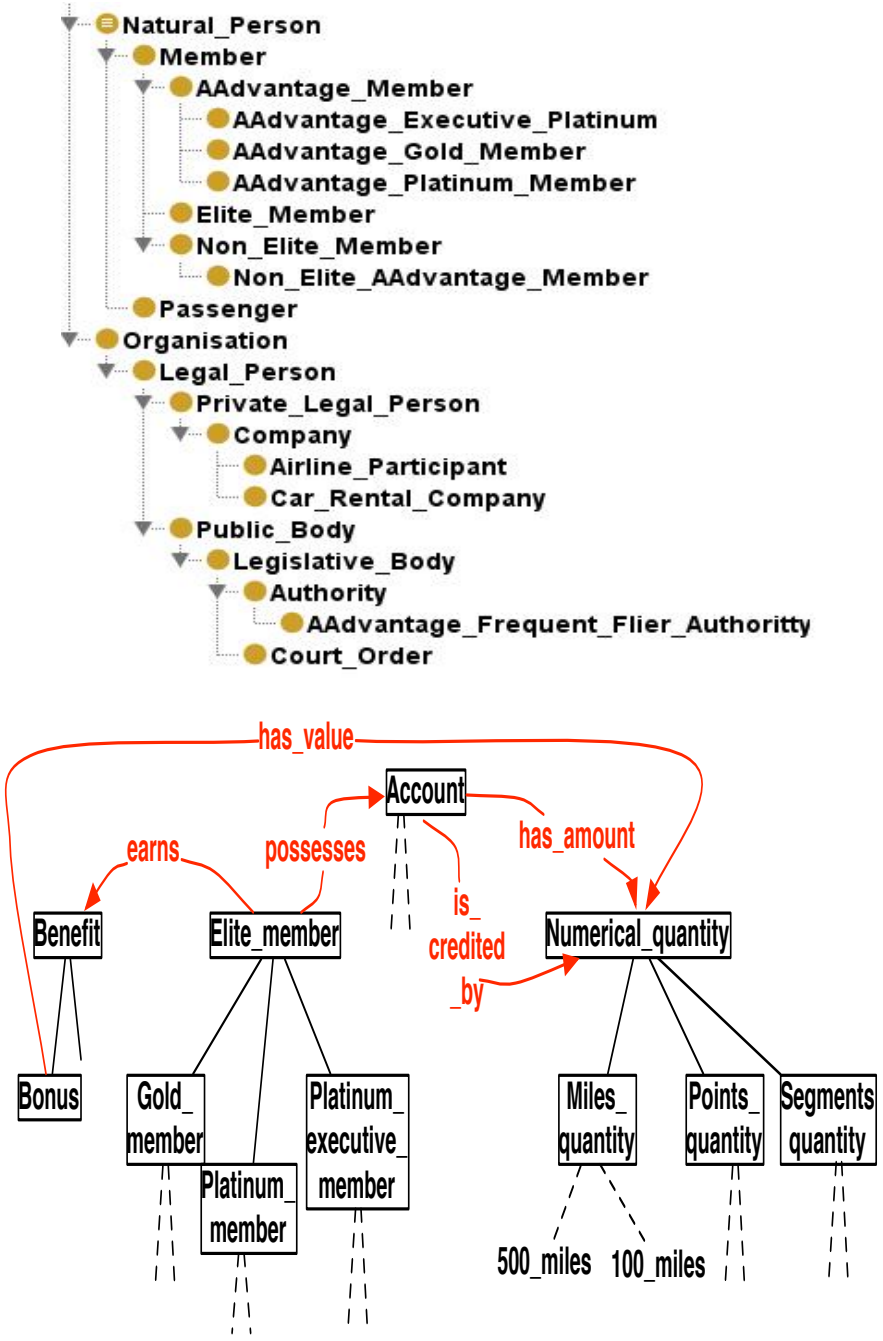



**Figure VI.2** – Exemple de deux fragments d'ontologies décrivant le vocabulaire utilisé dans AAdvantage (en haut la hiérarchie des concepts et au dessous les relations sémantiques entre concepts).

la propriété "**author**" qui précise l'expert métier auteur de la reformulation, et la propriété "**date**", la date de celle-ci.

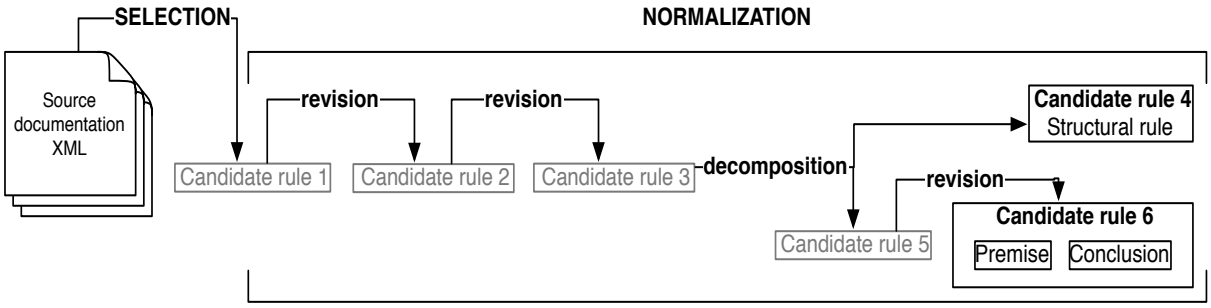



**Figure VI.3** – Processus de réécriture d'une règle candidate.


**Figure VI.4** – Exemple de réécriture d'une règle candidate.

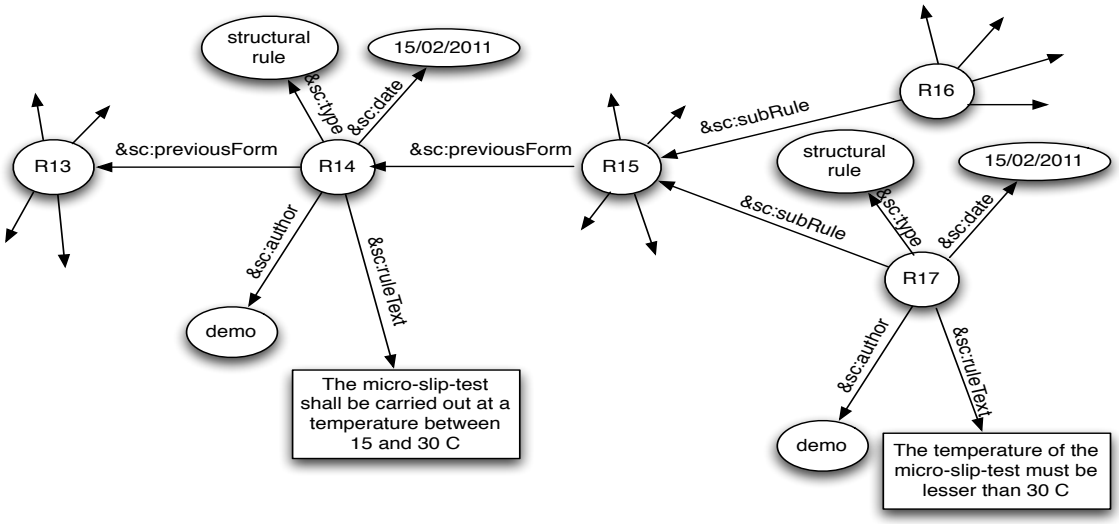



**Figure VI.5** – Graphe représentatif de la base des règles.

### VI.1.3 Modèle de liens

Le modèle de liens décrit un ensemble de couples $(ud_i, us_i)$ servant à faire la correspondance entre des unités documentaires $(ud_i)$ et des unités sémantiques $(us_i)$. Il décrit à la fois une connexion des textes à l'ontologie et aux règles candidates mais aussi une connexion des règles candidates à l'ontologie, soit tous les liens de la structure d'index. Autrement dit, il sert à connecter les termes métier apparaissant dans les textes et les règles avec les éléments conceptuels (concepts, relations, et individus) de l'ontologie et à connecter les règles candidates à leurs

fragments textuels (leurs phrases) où elles dérivent des textes d'origines.

Ce modèle se construit en projetant l'ontologie sur le texte pour l'annoter et au fur et à mesure que les règles candidates sont éditées à partir du texte, elles sont gardées connectées au texte par les fragments dont elles dérivent. Les liens entre les règles candidates et l'ontologie sont hérités des liens entre celle-ci et le texte et enrichis durant la normalisation. Le modèle est construit de façon graduelle, les premiers liens sont définis entre le texte et l'ontologie et ensuite chaque nouvelle règle candidate éditée définit de nouveaux liens.

Dans l'exemple de la figure VI.6, on voit un fragment de texte réglementaire où les liens sont visibles. Les couleurs représentent les termes métier associés aux concepts de l'ontologie (le vert souligné décrit les concepts, le rouge souligné décrit les individus instances de concepts, le bleu italique décrit les relations conceptuelles), et les règles candidates sont représentées sous forme d'ancres apposés sur le texte (les nombres entre crochets en début de certaines phrases montrent que celles-ci ont dérivé des règles candidates).

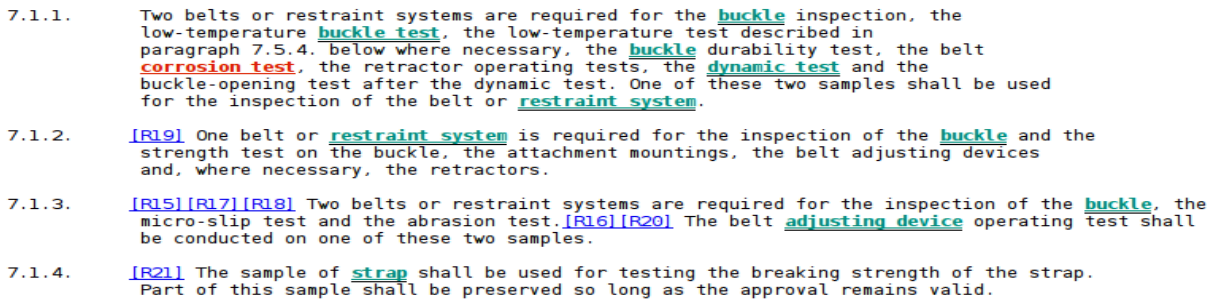



**Figure VI.6** – Un exemple de fragment de texte réglementaire annoté au regard d'une ontologie et d'une base de règles.

## VI.1.4 Représentation formelle de la structure d'index

Pour la mise en œuvre de la structure d'index, nous proposons une représentation formelle afin de réaliser la connexion du modèle de document au modèle sémantique à travers le modèle de liens. Pour ce faire, nous définissons un schéma (au sens des standards du Web) de la structure, et nous l'implémentons dans un graphe RDF.

### VI.1.4.1 Schéma de la structure

Un schéma définit un vocabulaire formel qui décrit tous les objets manipulés par la structure d'index ainsi que les relations qu'ils entretiennent. La figure VI.7 montre une représentation graphique de ce schéma dans laquelle nous pouvons voir qu'il définit trois types d'objets :

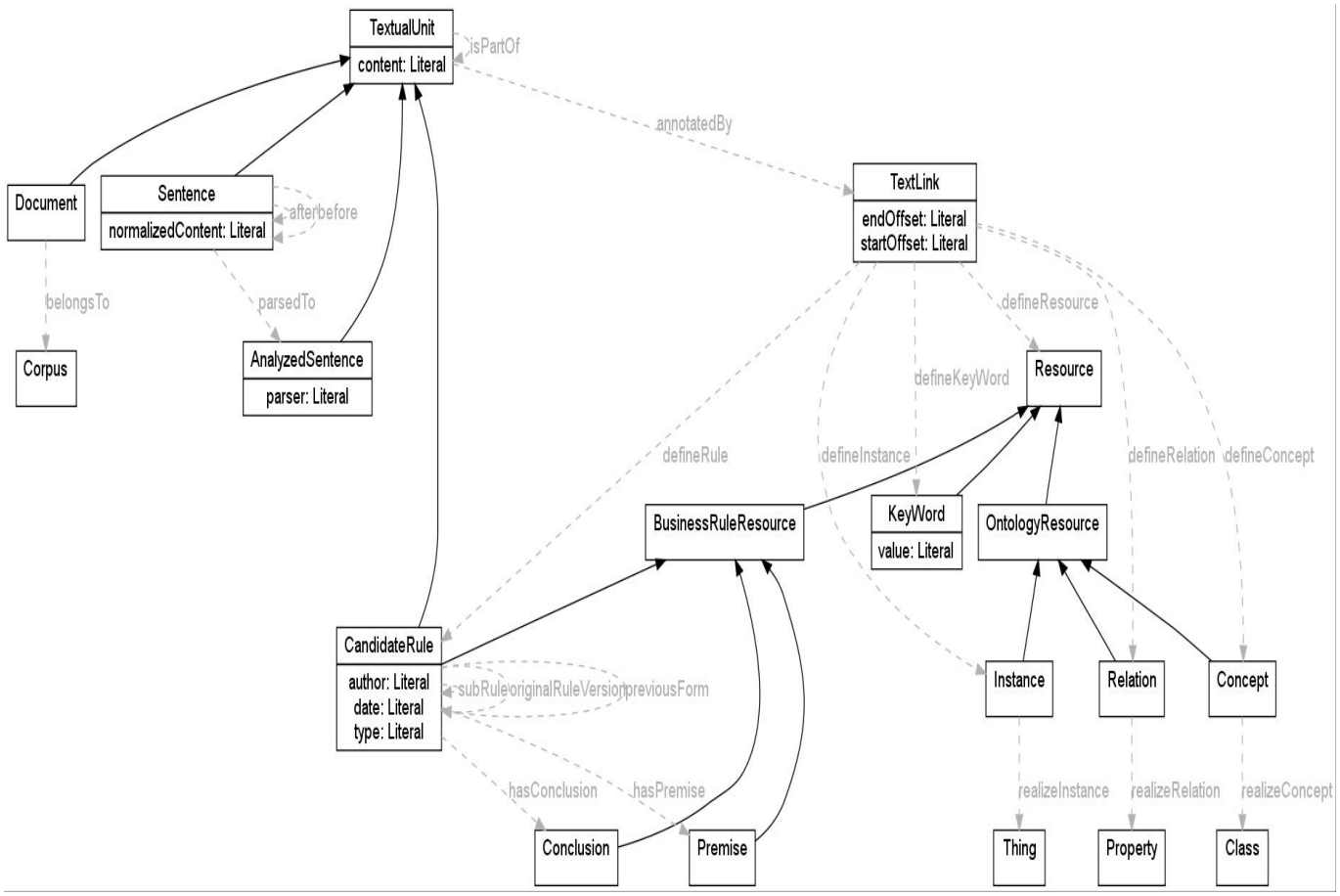



**Figure VI.7** – Schéma des liens.

**Les unités textuelles** ("**TextualUnit**") désignent les éléments du modèle de document. "**TextualUnit**" représente l'unité la plus basique à partir de laquelle toutes les autres sont construites. A une unité textuelle est associé un contenu (**content**) sous forme de chaîne de caractères. Nous ne considérons ici que les unités textuelles utiles au processus d'acquisition des règles à partir de texte et qui peuvent faire l'objet d'annotations. Il y a tout d'abord celles qui contiennent seulement des mots ou séquence de mots appartenant au vocabulaire métier de domaine ou aux mots-clés. Les unités structurelles sont plus spécifiques. Elles correspondent aux phrases ("**Sentence**") à partir desquelles les règles candidates sont dérivées et aux documents ("**Document**") desquels sont tirées les phrases. Un "**Document**" est contenu ("**belongsTo**") dans un "**Corpus**" de documents. Une unité textuelle peut en contenir une autre à travers la relation "**isPartOf**", cela permet de définir les dépendances entre les unités textuelles (par exemple nous pourrons dire qu'une "**TextualUnit**" est une partie d'une "**Sentence**"). Nous avons aussi

les unités "**AnalyzedSentence**" qui servent à contenir l'analyse syntaxique des phrases ("**parsedTo**") et sont utilisées dans la fouille syntaxique. Chaque "**AnalyzedSentence**" résulte d'un analyseur syntaxique dont le nom est donné par la propriété "**parser**". Enfin une règle candidate ( "**CandidateRule**") est aussi une unité textuelle, elle dispose d'un contenu textuel qui peut être annoté.

**Les ressources** ("**Resource**") désignent l'ensemble des objets utilisés pour annoter les unités textuelles. Il y a trois type de ressources, les ressources du modèle sémantique notamment l'ontologie ("**OntologyResource**") et les règles métier ("**BusinessRuleResource**"), et les mots-clés utilisés dans l'écriture des règles en langage contrôlé. L'objet "**OntologyResource**" définit tous les éléments conceptuels de l'ontologie ("**Concept**", "**Relation**" et "**Instance**") et l'objet "**BusinessRuleResource**" toutes les règles candidates ("**CandidateRule**") de la base de règles. Les concepts (respectivement les instances et les relations) définissent ("**realizeConcept**", "**realizeInstance**", "**realizeRelation**") des "**Class**" (respect. des "**Instance**" et des "**Property**") comme dans la plupart des modèles formels de représentation d'ontologies. Toutes les relations entre des règles candidates ainsi que les propriétés qui leurs sont associées sont aussi décrites dans le schéma ("**previousForm**", "**OriginalRuleVersion**", "**subRule**", "**ruleText**", "**type**", "**author**", "**date**"). En outre, chaque règle candidate peut être partitionnée en deux éléments à travers les relations "**hasPremisse**" et "**hasConclusion**", une "**Premisse**" qui spécifie sa condition d'application et une "**Conclusion**" qui spécifie l'opération à mettre en œuvre quant la prémisse est vérifiée.

**Les liens** ("**TextLink**") définissent les éléments du modèle de liens et décrivent les liens entre les objets **TextualUnit** et **Resource**. Un "**TextLink**" indique qu'une annotation définissant une unité sémantique ("**defineResource**") est apposée sur une unité textuelle ("**annotatedBy**"). Le lien peut aussi être typé en pointant sur un type de ressource précis ("**defineConcept**", "**defineInstance**", "**defineRelation**", "**defineRule**", et "**defineKeyWord**"). Et enfin le lien se localise dans le texte d'origine de l'unité textuelle à annoter, ce par les propriétés "**startOffset**" et "**endOffset**" qui spécifient le début et la fin de la chaîne de caractères de l'unité.

#### VI.1.4.2 Mise en œuvre du schéma dans un graphe RDF

Une implémentation de la structure d'index est nécessaire pour mettre en œuvre le schéma formel. Nous proposons une implémentation indépendante des méthodologies associées aux tâches d'acquisition et de maintenance des règles. Le but est d'avoir une implémentation qui

rende la structure de données commune à toutes les tâches et qui favorise son exploitation (voir VI.1).

Pour ce faire, nous avons fait le choix d'un graphe RDF (voir III.3.2.2) qui présente de réels avantages pour organiser des données et permettre la mise en place d'un cadre d'échange entre les différentes ressources manipulées. Nous implémentons le schéma sous forme de graphe RDF et tous les objets du schéma sont représentés dans des langages et technologies XML et du Web sémantique (voir III.3.2.1) compatibles avec RDF.

– **Un schéma en RDFs**. RDFs permet de définir des vocabulaires servant à organiser des données RDF. Il est utilisé ici pour décrire le schéma formel et organiser les ressources de la structure d'index. La figure VI.8 montre une partie du schéma utilisé. Tous les objets du schéma sont décrits sous forme de "**Class**" (exemple des classes "**TextualUnit**", "**CandidateRule**", "**TextLink**"). Les relations entre les objets (ainsi que les propriétés associées aux objets) sont de type "**Property**". La classe "**TextLink**" figure dans deux relations, la relation "**annotatedBy**" de la classe "**TextualUnit**" ("**rdfs :domain**") à la classe "**TextLink**" ("**rdfs :range**"), et la relation "**defineRule**" de la classe "**TextLink**" ("**rdfs :domain**") à la classe "**CandidateRule**" ("**rdfs :range**")

```
...
<rdfs:Class rdf:ID="TextualUnit"/>

<rdfs:Class rdf:ID="CandidateRule">
  <rdfs:subClassOf rdf:resource="#Resource"/>
</rdfs:Class>

<rdf:Property rdf:ID="annotatedBy">
  <rdfs:domain rdf:resource="#TextualUnit"/>
  <rdfs:range rdf:resource="#TextLink"/>
</rdf:Property>

<rdf:Property rdf:ID="defineRule">
  <rdfs:domain rdf:resource="#TextLink"/>
  <rdfs:range rdf:resource="#CandidateRule"/>
</rdf:Property>
...
```

**Figure VI.8** – Une partie du schéma formel de la structure d'index.

– **Un texte structuré en RDF**. Le schéma RDFs définissant une structure de document est le vocabulaire utilisé pour organiser les données textuelles selon cette structure. A chaque objet textuel (Document, Sentence, etc) est associé un identifiant unique sur l'ensemble de la structure d'index ("**rdf :ID**"). La figure VI.9 montre un fragment du document réglementaire AAdvantage sur lequel nous pouvons voir une structure du document en phrases ("**Sentence**").

```
<Document rdf:ID="#AAdvantageCorpus">
  ...
  <Sentence rdf:ID="#S11">
    <content>
    Each qualifying activity extends the expiration date
    of all unexpired mileage credit in your account for
    18 months from the date of the qualifying activity.
    </content>
  </Sentence>
  ...
</Document>
```

**Figure VI.9** – Structure d'arbre d'un texte réglementaire.

```
<!-- <!ENTITY onto ''http://lipn.univ-paris13.fr/RCLN/terminae/Audi'' -->

<owl:Class rdf:about="&onto;#AdjustingDevice">
    <rdfs:subClassOf rdf:resource="&onto;#Device"/>
</owl:Class>

<owl:Class rdf:about="&onto;#Space">
    <rdfs:subClassOf rdf:resource="&onto;#Thing"/>
</owl:Class>
```

---

```
<rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#AdjustingDevice">
        <skos:prefLabel>adjusting device</skos:prefLabel>
        <skos:altLabel>belt adjustment devices</skos:altLabel>
</rdf:Description>
<rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#Space">
        <skos:prefLabel>low-temperature chamber</skos:prefLabel>
        <skos:altLabel>refrigerated cabinet</skos:altLabel>
</rdf:Description>
```

**Figure VI.10** – Représentation OWL et SKOS des concepts "**AdjustingDevice**" et "**Space**".

– **Une ontologie en OWL enrichie d'un thésaurus en SKOS**. Ces deux ressources nous ont été fournies sous ces deux formats dans OntoRule. Elles ont été définies avec la plateforme TERMINAE[1]. TERMINAE fournit un thésaurus SKOS accompagnant l'ontologie OWL et dans lequel il associe à chaque concept ("**Class**") (respectivement à chaque instance ou à chaque relation) un ensemble de termes métier caractéristiques. A chaque fois qu'un de ces termes apparaît dans un texte réglementaire ou une règle, il est annoté par ce concept. L'un de ces termes est le terme préférentiel ("**prefLabel**"), le plus représentatif du concept, et les autres sont des termes synonymes ("**altLabel**") qui renvoient tous au même concept. Dans la figure VI.10, nous avons un extrait d'ontologie et un extrait du thésaurus, sur lesquels nous pouvons voir la définition des concepts "**AdjustingDevice**" et "**Space**" ainsi que les termes qui leurs sont associés. Chaque

1. http ://ontorule-project.eu/news/news/terminae

concept (respectivement instance ou relation) de l'ontologie représenté dans le thésaurus possède le même identifiant ("**rdf :ID**" ou "**rdf :about"**) dans le OWL et dans le SKOS.

– **Des règles candidates répertoriées dans un graphe RDF**. A partir du schéma de la structure d'index, l'ensemble des règles candidates forment une structure de graphe RDF suivant les relations de l'arbre des règles candidates ("**previousForm**", "**OriginalRuleVersion**", "**subRule**"). L'exemple de la figure VI.11 montre la description d'une règle candidate. L'identifiant (rdf :ID) de la règle candidate est "**R10**", elle est une révision de la règle "**R9**", la règle "**R6**" est sa première version identifiée et elle est une sous-règle de la règle "**R0**". Le contenu textuel de la règle ("**content**") est une chaîne de caractères transcrite en RDFa, afin d'y inclure tous ses liens avec l'ontologie.

If an safety_belt_assembly has a retractor then the length of strap has at least 300 +/- 3 mm.

```
<Rule rdf:ID="R10">

<ruleText>
<span rel="schema:annoted" typeof="schema:Rule"><span about="textlinkR10" rel="schema:defineResource" typeof="schema:TextLink"><span id="if" type="Keyword"
class="Keyword">If</span> <span id="an" type="Keyword" class="Keyword">an</span> <span rel="schema:realizeConcept" typeof="schema:Concept"><a about="http://lipn.univ-
paris13.fr/RCLN/terminae/Audi#SafetyBeltAssembly" class="Concept" href="javascript:void();" onclick="function1('SafetyBeltAssembly');">safety_belt_assembly</a></span>
has <span id="a" type="Keyword" class="Keyword">a</span> retractor <span id="then" type="Keyword" class="Keyword">then</span> <span id="the" type="Keyword"
class="Keyword">the</span> length <span id="of" type="Keyword" class="Keyword">of</span> <span rel="schema:realizeConcept" typeof="schema:Concept"><a about="http://
lipn.univ-paris13.fr/RCLN/terminae/Audi#Strap" class="Concept" href="javascript:void();" onclick="function1('Strap');">strap</a></span> has <span id="at"
type="Keyword" class="Keyword">at</span> <span id="least" type="Keyword" class="Keyword">least</span> 300 +/- 3 mm.</span></span>
</ruleText>
<date>19/11/2011</date>
<author/>
<type>Structural Rule</type>
<hasPremise/>
<hasConclusion/>
<previousForm rdf:resource="#R9"/>
<subRule rdf:resource="#R0"/>
<originalRuleVersion rdf:resource="#R6"/>

</Rule>
```

**Figure VI.11** – Exemple d'une règle candidate dans un graphe RDF.

– **Des annotations en RDFa**. Les annotations sont des liens entre les textes et les ressources sémantiques telles que l'ontologie ou la base de règles candidates. Nous proposons d'inclure ces liens dans les textes afin de faciliter la visualisation des annotations et l'exploration des textes. Pour ce faire, nous utilisons le format qui permet d'inclure des annotations RDF dans du HTML sous la base du vocabulaire défini dans le schéma. Dans l'exemple de la figure VI.12, nous avons un fragment de code RDFa renvoyant à l'annotation d'une phrase ("**sent0-578**") d'un texte au regard de son type dans le schéma ("**Sentence**"), et de concepts de l'ontologie ("**Buckle**") et d'une règle candidate ("**R16**") à travers un lien "**TextLink**" ("**textLink7**").

```
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
	<span about="http://lipn.univ-paris13.fr/RCLN/text/Audi#sent0-578" class="Sentence" id="sentence578" rel="schema:annoted" typeof="schema:Sentence">
		<span id="textlink7" rel="schema:defineResource" typeof="schema:TextLink">
Two belts or restraint systems are required for the inspection of the <span rel="schema:realizeConcept" typeof="schema:Concept"><a about="http://lipn.univ-paris13.fr/
RCLN/terminae/Audi#Buckle" class="Concept" href="#Buckle">buckle</a></span>, the micro-slip test and the abrasion test.</span><a class="Rule" about="http://lipn.univ-
paris13.fr/RCLN/ontorule/Audi/rules#R16" href="#R16">[R16]</a>
		</span>
	</span>
</body>
</html>
```

**Figure VI.12** – Exemple d'annotations intégrées à un fragment de texte.

### VI.1.5 Projection de l'ontologie sur un texte réglementaire

Le texte sur lequel travaille SemEX est déjà annoté au regard de l'ontologie. Pour produire ce texte en entrée, nous utilisons l'annotateur [2] développé au sein de l'équipe RCLN (développement auquel nous avons participé). L'annotateur permet de projeter les différents éléments ontologiques sur les termes métier présents dans le texte. Cette projection consiste à marquer dans le texte toutes les occurrences des termes métier synonymes associés à chaque concept (respectivement relation ou instance) par celui-ci. La recherche des occurrences prend en compte les variations morpho-syntaxiques des termes métier : l'annotateur les identifie à l'aide de leur lemmes.

L'annotateur est un plugin Java qui reçoit en entrée quatre fichiers :

- un fichier texte (.txt) correspondant au document à annoter ;
- un fichier .tt correspondant au résultat de l'analyse morphologique du document. Ce document est tabulaire et constitué de trois colonnes (une première pour les mots du document, une deuxième spécifiant les catégories morpho-syntaxiques de ces mots et une troisième spécifiant leurs lemmes) ;
- un thésaurus sous format SKOS (.rdf) contenant pour chaque concept (relation et instances) de l'ontologie la liste des termes métier qui lui sont associés ;
- une ontologie au format OWL.

Après projection, il fournit des annotations intégrées dans le texte au format RDFa, qui peuvent être visualisées dans un navigateur Web.

L'annotateur propose une interface utilisateur (voir figure VI.13) dans laquelle nous pouvons remplir un formulaire (fenêtre à gauche) pour charger les données en entrée et ensuite visualiser le résultat (fenêtre à droite). Le fichier produit peut être lu par SemEx qui charge ainsi la

2. http ://lipn.univ-paris13.fr/ szulman/Annotator/annotator.html
http ://lipn.univ-paris13.fr/ szulman/Annotator/annotator_manual.pdf

structure d'index initiale (sans aucune règle).

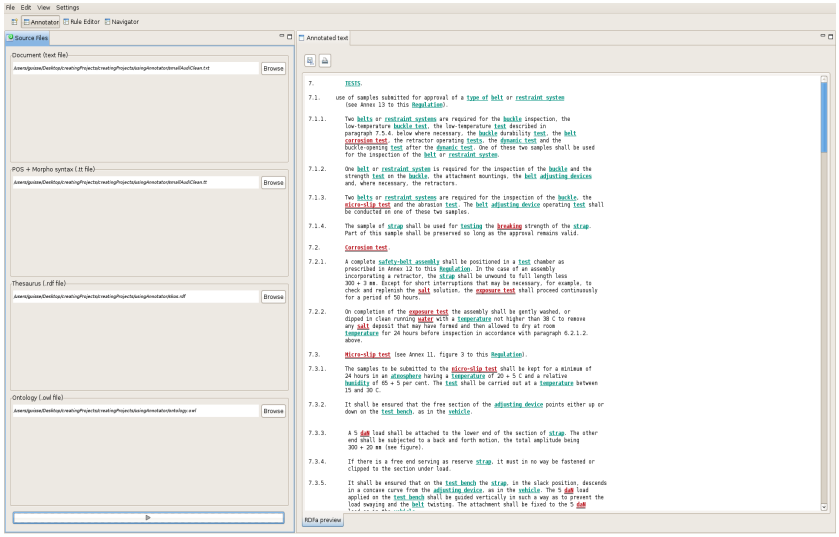


**Figure VI.13** – Interface utilisateur de l'annotateur de RCLN.

## VI.2 Conception de la plateforme SemEx

La figure VI.14 donne une vue d'ensemble de l'architecture de la plateforme SemEx. Elle montre l'organisation sur laquelle la plateforme s'appuie pour aider un expert métier à acquérir les règles à partir de textes réglementaires ainsi qu'à les maintenir.

Cette organisation résulte de l'interaction de quatre catégories hétérogènes d'éléments. Nous décrivons ici les utilisateurs et comment ils sont appelés à interagir avec SemEX, les différents modules qui mettent en œuvre les méthodologies définies plus haut (voir V) et les relations entre modules, les interfaces construites pour faciliter les méthodologies interactives. les données actualisées par chaque opération. Nous expliquons en même temps les flèches qui apparaissent sur la figure VI.14.

### VI.2.1 Les utilisateurs

En pratique, trois acteurs interviennent pour mettre en œuvre les règles métier dans un SGRM : l'expert métier chargé de rédiger les textes réglementaires, l'expert chargé de les réécrire dans une forme facile à formaliser, et l'ingénieur chargé de les formaliser et les automatiser. La plateforme SemEx est fournie pour guider le second acteur dans la réalisation de ses tâches.

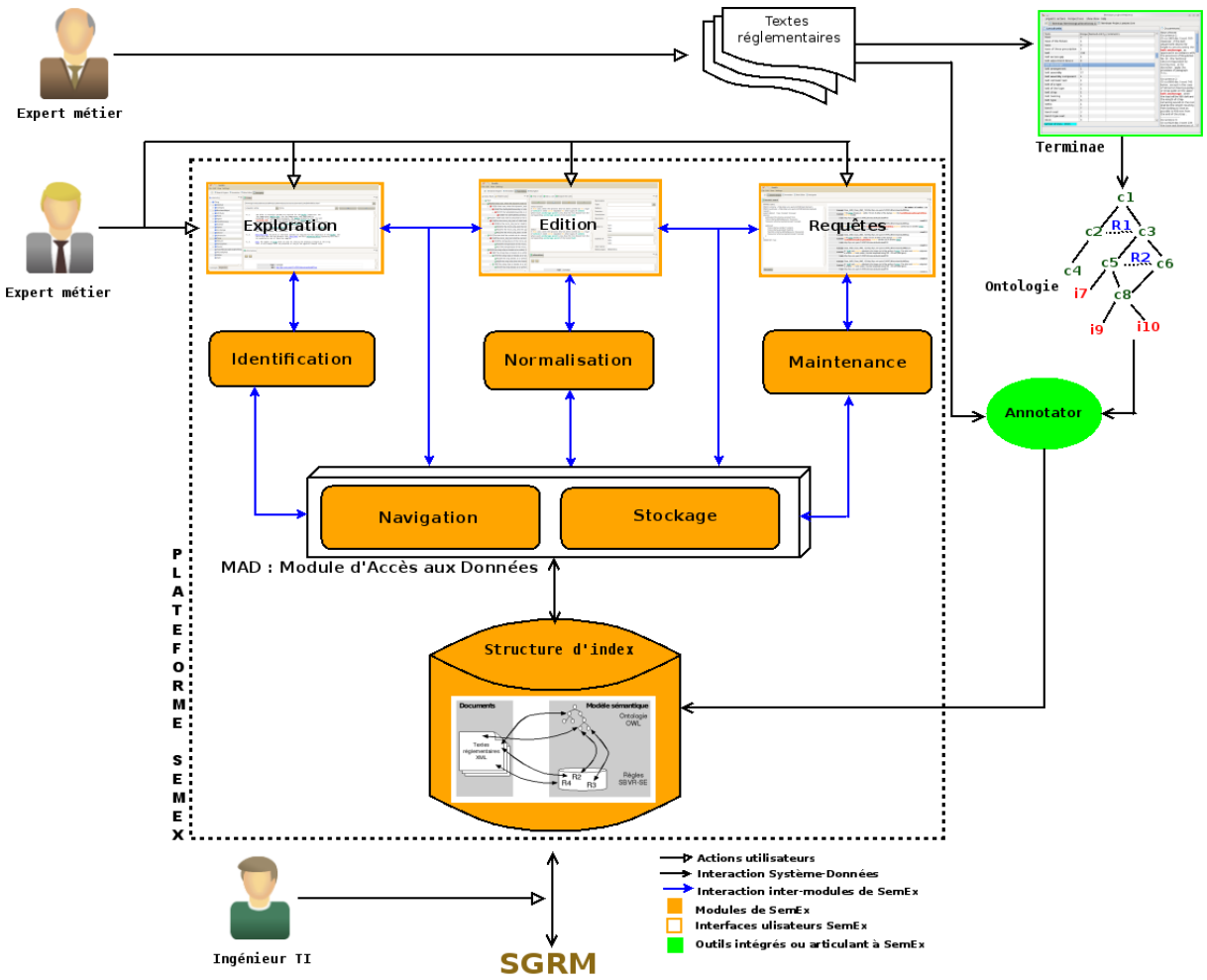



**Figure VI.14** – Vue d'ensemble de la plateforme SemEx.

L'expert métier utilise SemEx à travers trois interfaces interactives (voir la section VI.3) i) pour explorer et comprendre les textes, les fouiller pour y identifier les règles candidates, ii) pour éditer ces règles candidates dans le langage contrôlé SBVR-SE en leur appliquant des opérations de normalisation, iii) pour les maintenir et les garder en conformité avec les textes. En effet, l'expert peut interagir avec ces trois interfaces de façon indépendante. Il peut sélectionner manuellement ses règles candidates ou s'appuyer sur les fouilles automatiques, puis les sauvegarder dans la base de règles extraites. A tout moment il peut revenir vers les textes extraits et les visualiser pour ainsi passer à leur normalisation ou à leur maintenance.

### VI.2.2 Les modules

La plateforme repose sur un ensemble de modules qui implémentent les méthodologies d'acquisition et de maintenance des règles. Nous avons au plus haut niveau quatre modules dont trois (identification, normalisation, et maintenance) sont directement connectés aux interfaces utilisateurs et au quatrième module (lui-même composé) d'accès aux données. Les connections sont les suivantes :

- Le module identification est connecté à une interface d'exploration des textes afin de fouiller les règles candidates dans les textes.
- Le module de normalisation est connecté à une interface d'édition et permet de traduire les règles en SBVR-SE suivant la méthodologie de normalisation.
- Le module de maintenance sert à diagnostiquer les éventuels problèmes liés aux incohérences et à l'évolution des textes et à déceler les règles affectées à mettre à jour.
- Ces trois modules sont connectés au module d'accès aux données qui sert à manipuler toutes les ressources de la structure d'index. Il est composé de deux sous-modules, un module de navigation servant à interroger les données et un module de stockage des données (ajout des nouvelles ou modification ou suppression de anciennes). Comme son nom l'indique, ce module interagit avec tous les autres modules, il leur permet d'accéder aux données qu'ils manipulent. Il est aussi possible d'interagir avec ce module directement à partir des interfaces utilisateurs, notamment manipuler les données sans passer par les trois autres modules. Par exemple pour explorer les annotations conceptuelles (liens avec l'ontologie) d'un texte, il est possible d'interroger directement le module d'accès.

### VI.2.3 Les Interfaces

L'expert dispose dans SemEx des interfaces interactives pour le guider. Les interfaces incarnent les méthodologies mises en œuvre dans les modules. Chaque interface sert à visualiser les données et à leur associer des opérations dont les scénarios sont décrits par une méthodologie donnée. Une interface peut interagir avec un ou plusieurs modules (respectivement un module peut interagir avec une ou plusieurs interfaces) suivant l'opération déclenchée. Les interfaces permettent d'organiser les dépendances entre modules. Elles sont décrites plus en détail dans la section suivante (VI.3)

### VI.2.4 Les données

Elles correspondent aux ressources de la structure de données décrite en détail dans la section VI.1, notamment les textes, l'ontologie et les règles métier. Au début du processus, la plateforme Terminae permet de construire une ontologie lexicalisée autour du vocabulaire métier de domaine. Cette ontologie est projetée, à l'aide de l'outil Annotator, sur les textes (figure VI.14). On a donc une première version de l'index constituée des textes, de l'ontologie et des liens entre textes et ontologie. Ensuite au fur et à mesure que des règles candidates sont éditées l'index se trouve enrichi de nouveaux liens RC-ontologie et RC-textes. Ceci jusqu'à obtenir au final une base de règles normalisées en SBVR-SE et une structure d'index qui les garde connectées aux textes d'origine et à l'ontologie. Ces règles sont alors fournies aux ingénieurs pour qu'ils les formalisent.

## VI.3 Interfaces interactives

Nous avons construit actuellement trois interfaces qui sont décrites ci-dessous.

### VI.3.1 L'interface de navigation

Elle sert à faciliter l'exploration, l'analyse et la compréhension des textes réglementaires. Elle permet de naviguer dans les annotations entre les textes, l'ontologie et la base de règles, et aussi de fouiller les textes pour identifier les règles candidates. La figure VI.15, montre une fenêtre graphique représentative de cette interface qui se subdivise en quatre vues :

- Une vue (à gauche de la fenêtre) qui donne accès à l'ontologie par deux listes hiérarchiques. La première liste ses concepts (icône 'C') et leurs instances (icône 'I'), la seconde liste ses relations conceptuelles.
- Une vue (à droite de la fenêtre) montre les textes dans lesquels sont intégrées des annotations décrivant les liens avec l'ontologie et la base de règles. Les termes d'un texte renvoyant à des éléments conceptuels sont colorés [3] (vert pour les concepts, rouge pour instances et bleu pour les relations) et les règles candidates extraites sont représentées sous forme d'ancres (l'identifiant de la règle candidate entre crochets) devant les phrases à partir desquelles elles ont été dérivées. En outre, il est possible pour l'expert de sélectionner une phrase et de l'envoyer en tant que règle à l'interface d'édition.

3. Les couleurs sont inspirés des styles utilisés de SBVR

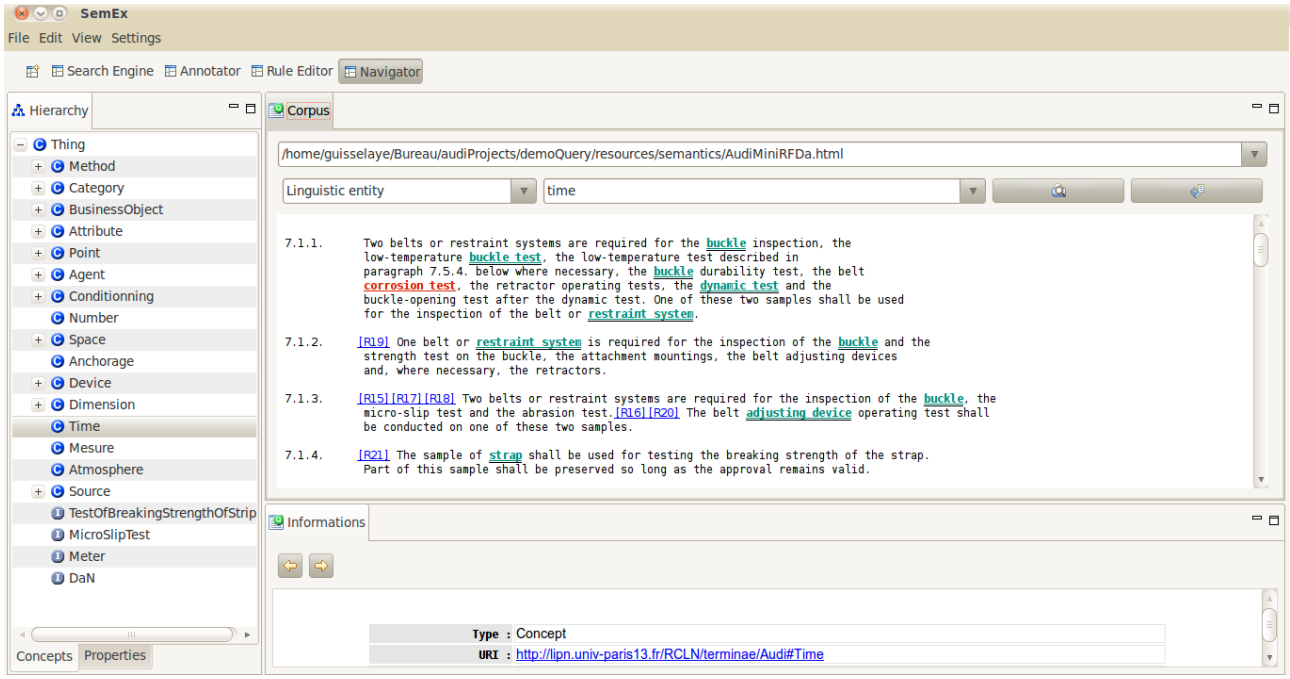



**Figure VI.15** – Interface de navigation.

- Une vue (en bas de la fenêtre) qui visualise les détails de certains éléments figurants dans les deux vues précédentes. Elle présente des informations rattachées à un concept dans l'ontologie ou à une annotation dans le texte (qui est donc aussi associée à un concept). A chaque clic sur un de ces éléments, la vue est automatiquement chargée des informations rattachées à l'élément. Il s'agit des données contextuelles du concept dans l'ontologie notamment son père, ses frères, ses instances, ses termes métier synonymes associés, etc, mais aussi toutes les autres phrases du texte et toutes les règles candidates de la base de règles où il apparaît. Cette vue est aussi accessible dans l'interface d'édition pour visualiser les informations rattachées aux annotations des règles candidates au regard de l'ontologie.
- Une vue (à droite de la fenêtre dans un deuxième onglet) de fouille des règles candidates présentes dans le texte. Dans cette vue, l'expert a la possibilité de choisir des mots-clés SBVR ou concepts et de visualiser toutes les phrases filtrées par ces derniers. Sur la vue, nous pouvons voir la liste de ces phrases que nous pouvons sélectionner une à une pour les envoyer en tant que règles à l'interface d'édition.

### VI.3.2 L'interface d'édition

L'interface d'édition de règles est celle à partir de laquelle les règles sont normalisées, stockées dans la base de règles et visualisées. Elle est constituée de trois vues (voir la figure VI.16) :

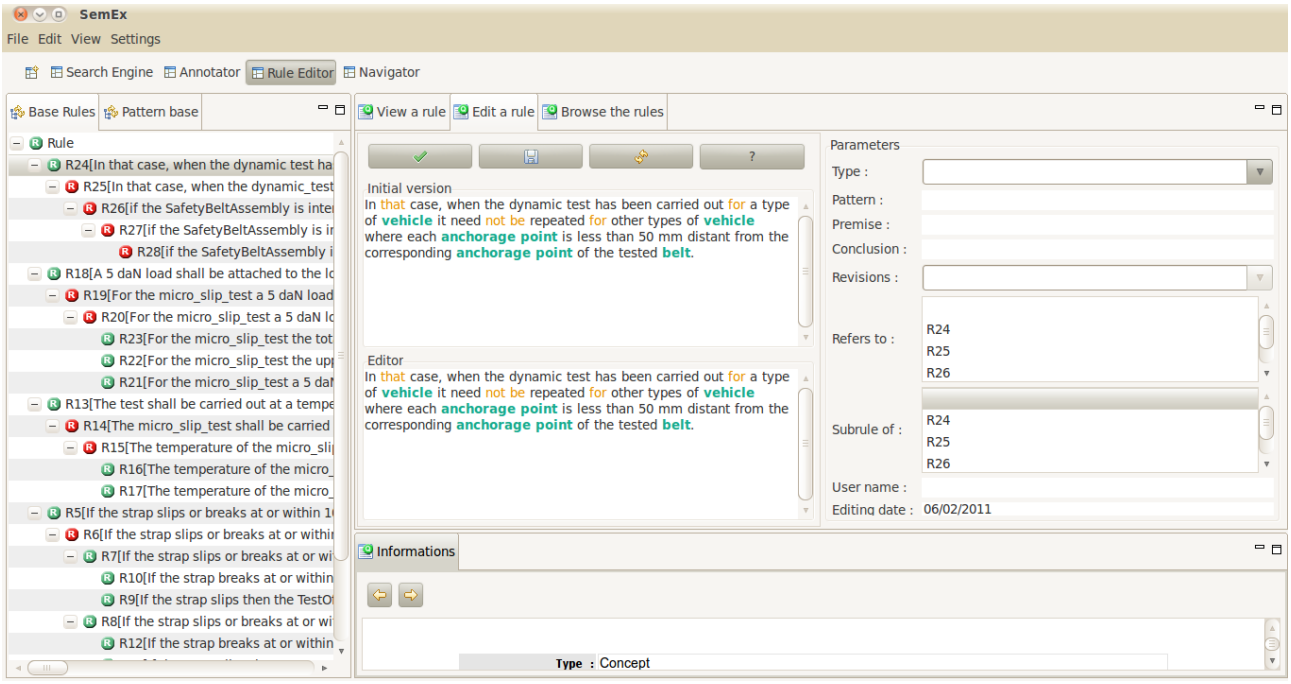



**Figure VI.16** – Interface d'édition des règles candidates.

- Une vue (à gauche de la fenêtre) qui présente la base de règles sous une forme d'un arbre montrant l'historique des reformulations subies par chacune des règles candidates. Les reformulations résultant d'une division sont représentées en vert, et celles résultant d'une révision sont en rouge. Le choix des couleurs ne répond à aucun critère, il est libre et sert à différencier les versions.
- Une vue (à droite de la fenêtre) qui visualise une règle candidate dans sa forme annotée, ainsi que toutes ses informations associées notamment ses propriétés, sa version originale dans le texte d'avant reformulation, sa version précédente et ses versions suivantes. Elle est connectée à la vue précédente : à chaque fois qu'un clic est effectué sur une règle candidate de l'arbre des reformulations, cette vue ci visualise la règle en question.
- Une vue (dans un deuxième onglet à droite de la fenêtre) qui contient le formulaire de normalisation d'une règle candidate en SBVR-SE. Le formulaire est constitué d'un champs qui montre la règle candidate qui est en train d'être éditée, un champs d'édition de la règle candidate et un ensemble de champs qui permettent de paramétrer la règle et qui correspondent à ses propriétés et à ses relations avec les bases de règles. Le champs

d'édition est muni d'une auto-complétion qui reconnaît automatiquement les mots clés SBVR-SE ainsi que les termes métier. C'est à partir de cette vue qu'une règle est sauvée dans la base, est modifiée, est supprimée.

– Une vue (en bas de la fenêtre) qui visualise les détails des annotations, la même que dans l'interface de navigation. Cette vue est renseignée chaque fois que l'on clique sur une annotation de la règle candidate dans la vue détaillée des règles.

### VI.3.3 L'interface de requêtes

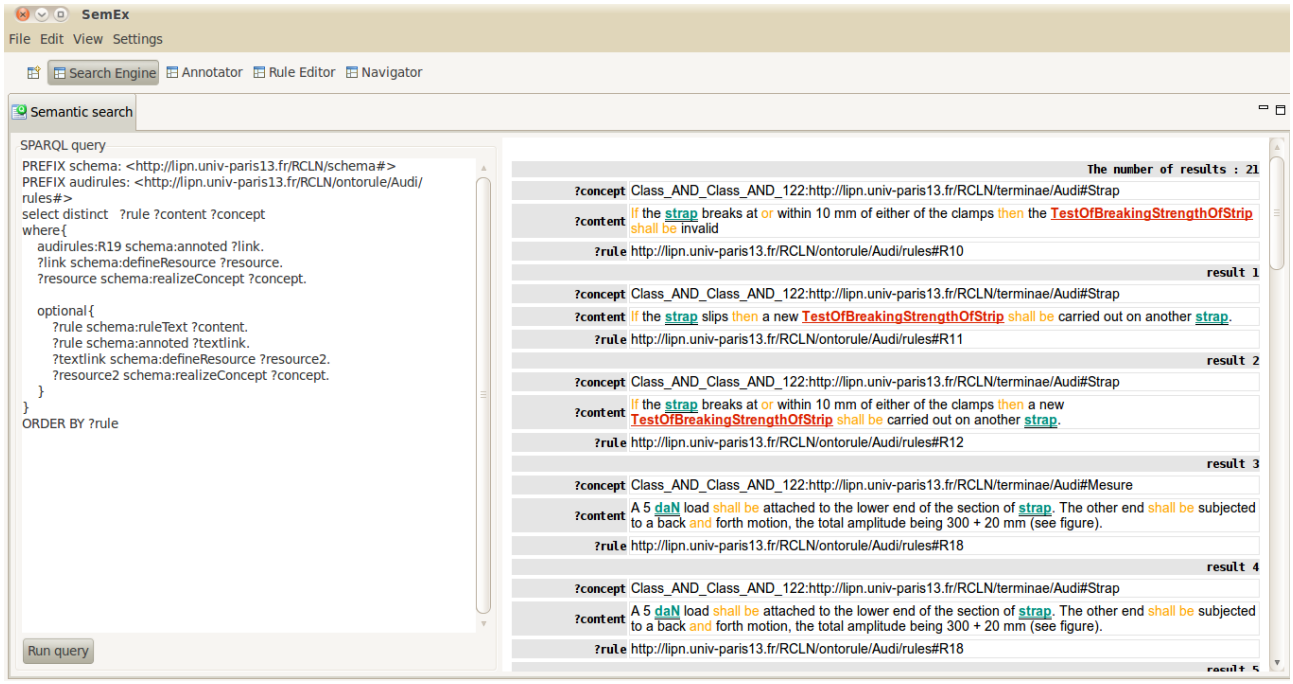



**Figure VI.17** – Interface de requêtes.

Elle permet d'explorer la structure d'index et de naviguer dans son graphe RDF. C'est une interface de requêtes SPARQL qui sont exécutées sur le graphe. Elle est constituée de deux vues, une vue d'édition de requête et une vue qui montre les résultats. Toutes les requêtes applicables sur le graphe peuvent être utilisées sur cette interface (voir la section VI.4.1) mais elle a surtout été proposée pour aider l'expert dans le diagnostic des problèmes de maintenance qui l'aideront à mettre à jour la base de règles (voir VI.7). Dans ce cas, il faut revenir sur l'interface d'édition pour modifier les règles avant de les sauver.

## VI.4 Module d'accès aux données

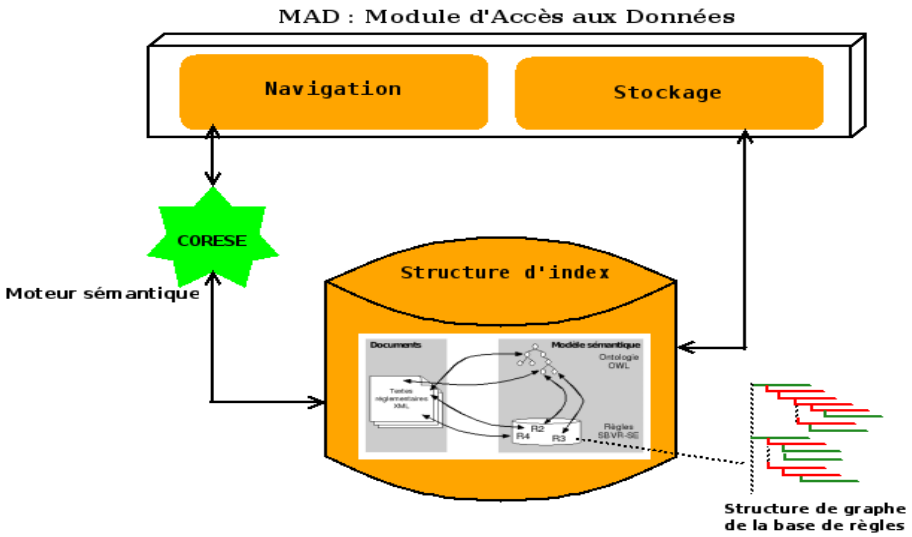



**Figure VI.18** – Module d'accès aux données.

Ce module (voir la figure VI.18 permet l'accès aux ressources manipulées soit pour naviguer dans la structure d'index soit pour stocker de nouvelles règles candidates dans la base de règles. Il est ainsi composé de deux sous-modules qui sont le module de navigation et le module de stockage. Dans le premier, l'objectif est d'implémenter les accès possibles et utiles à nos tâches. Le module de navigation communique avec les modules d'identification, de normalisation et de maintenance ainsi qu'à toutes les interfaces utilisateurs. Le second module sert à placer une nouvelle règle candidate dans la structure de la base de règles et à mettre à jour les liens de l'index. Ce sous-module communique uniquement avec l'interface d'édition des règles, il est chargé de sauver toute règle normalisée. Toutes les modifications et suppressions de règle candidate sont aussi prises en charge dans ce sous-module.

### VI.4.1 Navigation

La navigation consiste à interroger la structure d'index afin d'exploiter l'espace de liens qui la composent. Nous visons une exploitation utile à nos tâches, à la fois pour l'exploration sémantique des textes, pour faciliter leur fouille, la normalisation et la maintenance des règles candidates. L'idée est surtout d'exploiter le graphe RDF obtenu et de définir dessus un système de requêtes (voir Figure VI.19) qui intègre toutes les fonctionnalités de l'espace de liens. Nous décrivons ici les principales fonctionnalités pour lesquelles nous proposons une implémentation. Nous découvrirons l'usage de ces fonctionnalités dans l'implémentation de chacun des modules.

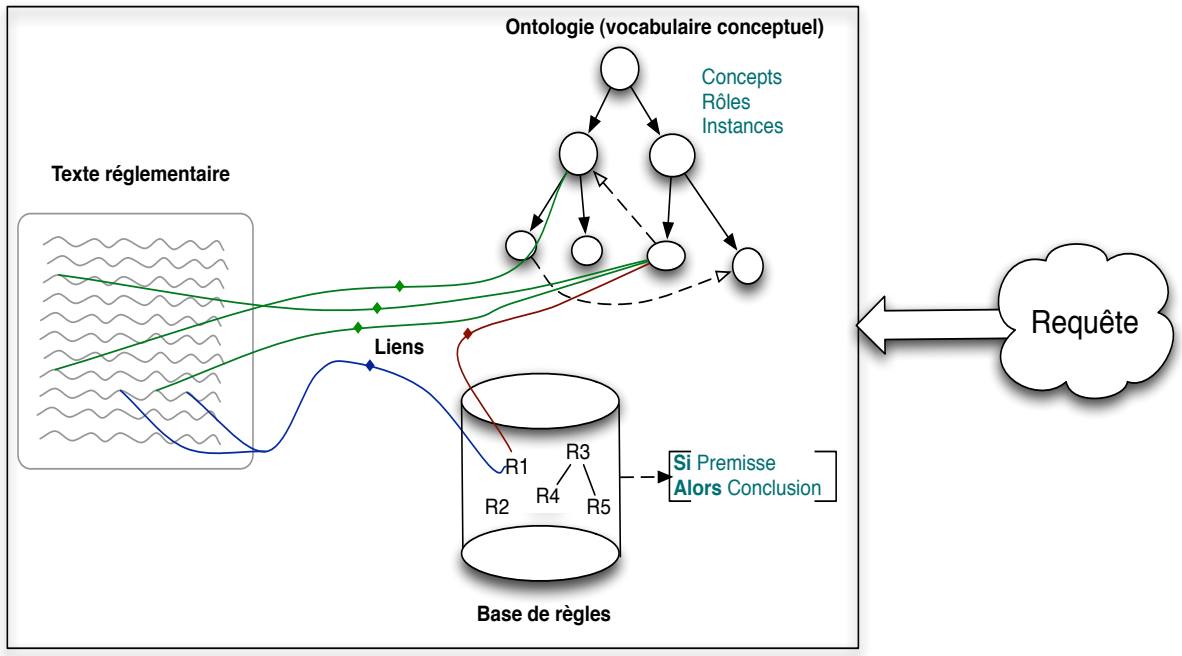



**Figure VI.19** – Navigation dans la structure d'index.

### VI.4.1.1 Types de navigation

La structure d'index définit un espace à partir duquel nous proposons deux types de navigations :

**Navigation intra-ressource** .

Elle définit les accès dans une même ressource. La navigation consiste à articuler les différentes unités d'une ressource :

- Les unités du modèle documentaire (**TextualUnit**, **Sentence**, **Document**, etc). Cette navigation permet d'articuler directement une unité textuelle à une autre comme par exemple pour retrouver des phrases d'un document donné, les occurrences d'un terme métier dans une unité supérieure (une phrase, un document, . . . ), toutes les phrases qui contiennent deux termes métier donnés, etc. Il est possible de naviguer de façon moins directe, par exemple pour retrouver les phrases analysées d'un document, il faudra passer par les phrases du document qui ont été analysées ("**ParsedTo**").
- Les unités de l'ontologie (**Concept**, **Relation**, **Instance**). Ici la navigation est classique pour une ontologie, il s'agit de passer d'un élément à un autre sur la base de la hiérarchie conceptuelle, des relations sémantiques, des logiques associées, etc. Cela permet de mettre en œuvre une variété de recherches pour retrouver entre autre les

instances d'un concept, les termes métier synonymes d'un concept, les concepts pères à un concept donné, etc.
- Les unités de la base de règles qui sont les **CandidateRule**. Il s'agit de naviguer dans la structure de graphe de la base de règles qui enregistre toutes les reformulations. Cela permet alors de retrouver l'historique d'une règle candidate donnée, de retrouver la version finale d'une règle candidate à partir de sa version originale, de retrouver une règle candidate à partir de caractéristiques tirées de la base de règles, etc.

**Navigation inter-ressources** .

Ce type de navigation définit une articulation de ressources différentes. Elle repose sur une exploration des liens formés par la structure d'index entre :

- **Les textes et l'ontologie**. On passe d'unités textuelles à des unités sémantiques tirées de l'ontologie (concepts, instances, relations) et inversement. Une telle navigation permet de retrouver par exemple les concepts annotateurs d'une phrase donnée ou vice versa les phrases qui ont été annotées au regard d'un concept donné, etc.
- **Les textes et la base de règles**. On passe d'unités textuelles à des unités sémantiques tirées de la base de règles (règles candidates) et inversement. Une telle navigation permet de retrouver par exemple les règles candidates dérivées d'une phrase donnée, la phrase ou le contexte d'une phrase (les unités textuelles qui l'entourent) à partir de laquelle une règle a été dérivée, etc.
- **La base de règles et l'ontologie**. On navigue entre les unités sémantiques, c'est-à-dire entre les éléments conceptuels et les règles candidates. Une telle navigation permet de retrouver les règles candidates dans lesquelles apparaît un concept donné ou vice versa les concepts ayant servi à annoter une règle candidate donnée.
- **Toutes ces trois ressources à la fois**. On peut circuler entre des unités de ressources différentes tout en faisant intervenir une troisième ressource. Une telle navigation permet de retrouver les concepts annotateurs des règles candidates dérivées d'une phrase donnée, les règles candidates dans lesquelles occurrent les mêmes concepts qu'une phrase donnée ou vice versa les phrases dans lesquelles occurrent des concepts utilisés pour annoter une règle candidate donnée, les concepts annotateurs d'une phrase et qui apparaissent ou qui disparaissent durant les reformulations des règles candidates dérivées, etc.

#### VI.4.1.2 Moteur sémantique et requêtes SPARQL

La mise en œuvre de l'espace de navigation consiste à mettre en place un système de requêtes implémentant toutes les fonctionnalités de recherche prévues. Comme il s'agit d'exploiter un graphe RDF, les requêtes sont implémentées dans un langage dédié, le SPARQL. Nous présentons dans l'annexe (voir A) des exemples de requêtes SPARQL qui permettent d'articuler les textes, l'ontologie et la base de règles candidates. Les requêtes SPARQL sont définies sur la base du schéma formel, avec le vocabulaire de celui-ci. Par exemple sur la figure VI.20 nous avons une requête permettant de retrouver les phrases (variables de type **Sentence**) annotées par (**schema :annotatedBy**) un lien ( **?textlink**) pointant sur une ressource conceptuelle (**schema :defineConcept**) qui soit est inconnue ( grâce au `optional`), soit désigne (schema :realizeConcept) le concept "**onto :Mileage**".

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX onto: <http://lipn.univ-paris13.fr/RCLN/terminae/AAdvantage#>

select ?sentence
where
{
 ?sentence rdf:type schema:Sentence.
 ?sentence schema:annotatedBy ?textlink.
 ?textlink schema:defineConcept ?resource.
 optional{?resource schema:realizeConcept "onto:Mileage"}
}
```

**Figure VI.20** – Exemple de requête SPARQL.

Pour l'application des requêtes SPARQL sur le graphe RDF, il est nécessaire d'utiliser un moteur sémantique comme nous le décrivons dans III.3.2.2. Nous utilisons le moteur CORESE, un logiciel libre à code public dédié à la projection de requêtes SPARQL sur des graphes RDF. Pour ce faire, nous chargeons le graphe de la structure d'index dans CORESE, qui prend en entrée des données RDF et produit des réponse au format XML que nous pouvons manipuler facilement dans les autres modules et interfaces de SemEx.

En pratique notre graphe RDF correspond à un ensemble de fichiers RDF, structurés autour du schéma de la structure d'index :

- Un fichier OWL (.owl) implémentant l'ontologie,
- Un fichier SKOS (.rdf) implémentant le thésaurus associé à l'ontologie,
- Un fichier RDF (.rdf) implémentant le modèle documentaire,
- Un fichier RDF (.rdf) implémentant le graphe de la base de règles candidates,

– Et un fichier RDF (.rdf) correspondant au graphe RDF inclus dans le fichier RDFa (.xhtml) implémentant les liens de l'index. L'extraction de ce graphe à partir du RDFa s'effectue suivant le mécanisme appelé GRDDL [4] permettant d'extraire le contenu RDF à partir d'un contenu XML ou XHTML, en y appliquant un fichier XSLT.

### VI.4.2 Stockage des règles

La base de règles est décrite dans une structure de graphe RDF. L'ensemble des règles candidates décrit un ensemble de triplets RDF qui correspondent aux caractéristiques (**type**, **date**, **author**, **hasPremise**, **hasConclusion**, etc) des règles candidates ainsi qu'aux relations qu'elles entretiennent (**previousForm**, **subRule**, **originalRuleVersion**). Ces dernières relations permettent d'obtenir une base sous forme d'arbre résumant la hiérarchie entre les règles candidates qui résulte des reformulations et des décompositions effectuées. Le sous-module de stockage permet alors de placer chaque nouvelle règle candidate dans la base de règles et de mettre à jour l'espace de liens de la structure d'index.

Techniquement la base de règles est un fichier RDF dans lequel les nouvelles règles candidates sont ajoutées les unes après les autres. Une seule relation de descendance suffit théoriquement pour organiser les règles dans une structure d'arbre. Notre représentation y ajoute quelques redondances pour pallier les limitations de SPARQL. Tout d'abord, la descendance est notée différemment selon qu'il s'agit d'une révision ou d'une division. Nous utilisons la relation (**previousForm**) pour la révision : la règle mère n'a qu'un seul descendant qui lui est sémantiquement équivalent ; nous disons que la mère est la version précédente. La relation **subRule** note une division ; la règle mère a plusieurs descendants et est sémantiquement équivalente à leur conjonction ; nous parlons alors de sous-règle. Différencier les deux cas évite des requêtes pour déterminer si une règle particulière est objet de plusieurs triplets de la relation de descendance, requêtes qui doivent parcourir tous les triplets.

Pour constituer l'arbre, nous définissons une règle candidate abstraite "R0" comme sa racine et toutes les premières versions de règles candidates sont placées comme sous-règle de R0. Ensuite, d'autres redondances sont introduites qui remplacent des appels récursifs aux relations de descendance directe [5].

– Toute règle candidate sauf R0 est sous-règle d'une autre règle. Soit la règle candidate résulte directement d'une division et elle est sous-règle par construction, soit elle résulte

4. Gleaning Resource Descriptions from Dialects of Languages (http ://www.yoyodesign.org/doc/w3c/grddl-primer/ en français et http ://www.w3.org/TR/2006/WD-grddl-primer-20061002/ en anglais)

5. Sparql n'implémenta pas ces appels récursifs, car le graphe RDF auquel il les appliquerait n'est pas garanti acyclique.

```
−<rdf:RDF xml:base="http://lipn.univ-paris13.fr/RCLN/ontorule/Audi/rules">
   <CandidateRule rdf:ID="R0"/>
  −<CandidateRule rdf:ID="R1">
      <content/>
      <date/>
      <author/>
      <type/>
      <hasPremise/>
      <hasConclusion/>
      <subRule rdf:resource="#R0"/>
    </CandidateRule>
  −<CandidateRule rdf:ID="R2">
      ...
      <previousForm rdf:resource="#R1"/>
      <subRule rdf:resource="#R0"/>
      <originalRuleVersion rdf:resource="#R1"/>
    </CandidateRule>
  −<CandidateRule rdf:ID="R3">
      ...
      <previousForm rdf:resource="#R2"/>
      <subRule rdf:resource="#R0"/>
      <originalRuleVersion rdf:resource="#R1"/>
    </CandidateRule>
  −<CandidateRule rdf:ID="R4">
      ...
      <previousForm rdf:resource="#R3"/>
      <subRule rdf:resource="#R0"/>
      <originalRuleVersion rdf:resource="#R1"/>
    </CandidateRule>
  −<CandidateRule rdf:ID="R5">
      ...
      <previousForm rdf:resource="#R4"/>
      <subRule rdf:resource="#R0"/>
      <originalRuleVersion rdf:resource="#R1"/>
    </CandidateRule>
  </rdf:RDF>
```

**Figure VI.21** – Structure de graphe RDF de la base de règles candidates.

d'une révision et elle est notée sous-règle de la même règle que sa mère.

- Les sous-règles directes (résultant directement d'une division) se distinguent à ce qu'elles n'ont pas de relation **previousForm**.
- Toute règle candidate, hormis les versions originales elles-mêmes, sont gardées connectées à leur version originale, autrement dit la version qui est sous-règle de R0, par la relation **originalRuleVersion** héritée par tous les descendants.

Dans l'exemple de la figure VI.21, nous pouvons voir la définition de cinq règles associées à la règle abstraite "R0". La règle "R1" est la seule dérivée de "R0", elle est une version originale de règle, la règle "R2" est la révision de "R1" et donc sous-règle de "R0", sa version originale

est ”R1”, etc.

Chaque règle candidate de la base dispose d'un identifiant unique qui la différencie des autres. Cet identifiant est déterminé par un entier qui s'incrémente au fur et à mesure que des règles sont rajoutées.

Une fois la première version de la règle placée dans la base de règles, l'espace de navigation est aussi mis à jour. Pour cela, une ancre avec son identifiant est placée dans le texte au début de la phrase dont elle a été dérivée. Cela constitue une connexion de la version originale et de son texte d'origine. Le contenu textuel de chaque version de règle candidate est quant à lui annoté au regard de l'ontologie et traduit au format RDFa.

En plus des ajouts de nouvelles règles candidates, la base de règles peut subir d'autres types de mises à jour durant sa construction. Il s'agit notamment de la modification et de la suppression de règles candidates existantes. Dans le premier cas la règle candidate ne change pas d'identifiant, seules ses propriétés et ses relations avec les autres règles candidates sont modifiées. Son lien avec sa version d'origine ne change pas mais les annotations de son contenu textuel peuvent changer. Dans le second cas, quant une règle candidate est supprimée de la base, son contenu est supprimé donc elle n'a plus de lien avec l'ontologie, tous les triplets RDF où elle figure (comme sujet ou comme objet) sont supprimés, son lien avec le texte est aussi supprimé.

## VI.5 Module d'identification

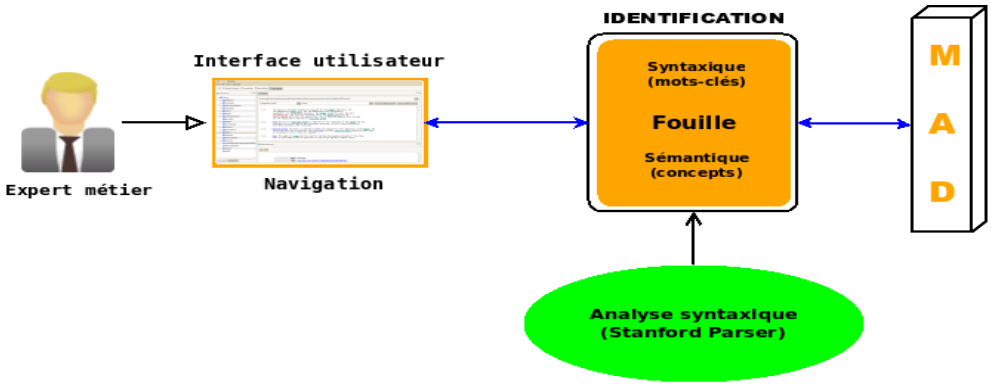



**Figure VI.22** – Module d'identification des règles candidates à partir des textes.

Le module d'identification implémente la méthodologie d'identification des règles candidates (V.2) à partir des textes. Il met en œuvre les techniques de fouille qui guident l'expert métier dans cette tâche. L'idée est d'avoir un module qui prenne en charge toutes les phases

automatisables de la tâche. En effet, le module (voir figure VI.22) interagit avec l'interface de navigation à partir de laquelle l'expert choisit les mots-clés caractéristiques de règles ou les termes métier pouvant décrire des règles et le module est chargé de lancer automatiquement les fouilles par mot-clé et sémantique. L'automatisation de ces fouilles repose sur la définition des patrons linguistiques et consiste à les appliquer sur la base documentaire pour identifier toutes les phrases pertinentes filtrées par les patrons.

Le module d'identification interroge aussi le MAD pour l'accès aux données qu'il manipule. Celles-ci sont les phrases de la base documentaire, les mots-clés et les annotations ontologiques.

## VI.5.1 Les données du module d'identification

### VI.5.1.1 Analyse et représentation des phrases

Les phrases renvoient aux objets "**Sentence**" de la structure d'index. Elles sont utilisées selon le type de fouille qui est effectuée. Dans la fouille syntaxique, les patrons sont appelés à matcher avec le contenu brut normalisé d'une phrase ou son contenu analysé syntaxiquement. Le premier cas est pris en charge par le moteur de requête SPARQL (CORESE) qui utilise les propriétés de l'objet "**Sentence**". Pour ce qui est du second, nous avons mis au point son schéma, mais l'implémentation reste à faire. Le contenu analysé d'une phrase correspond à son arbre syntaxique après passage d'un analyseur syntaxique. Nous nous sommes appuyés pour les structures de données sur le **Stanford Parser** qui fournit à la fois une analyse en composants (voir figure VI.23) et une analyse des dépendances de la phrase. Le Stanford Parser [6] est souvent décrit (comme dans [Tobon and Franco, 2010]) comme un analyseur complet et pratique mais nous l'avons choisi parce que son implémentation disponible en Java est facilement articulable avec SemEx. Dans le schéma de la structure d'index (VI.7) est défini l'objet "**AnalyzedSentence**" dans lequel nous stockons l'analyse d'une phrase donnée. Il est pointé par la relation "**parsedTo**" depuis l'objet "**Sentence**" qui contient la phrase brute. L'objet "**AnalyzedSentence**" a deux propriétés : "**content**", hérité de l'objet "**TextualUnit**", qui permet de stocker l'arbre syntaxique sous une forme parenthésée (fournie pour l'instant par le Stanford Parser) et la propriété "parser" notant le nom de l'analyseur. La figure VI.24 montre comment une phrase est alors stockée dans le schéma. Ce dispositif nécessite aussi d'intégrer les appels aux outils Tregex et Tsurgeon (expression régulière d'arbres) dans le module d'accès aux données. La difficulté principale est toutefois l'adaptation de l'analyseur à la langue du domaine.

6. http ://nlp.stanford.edu/software/lex-parser.shtml

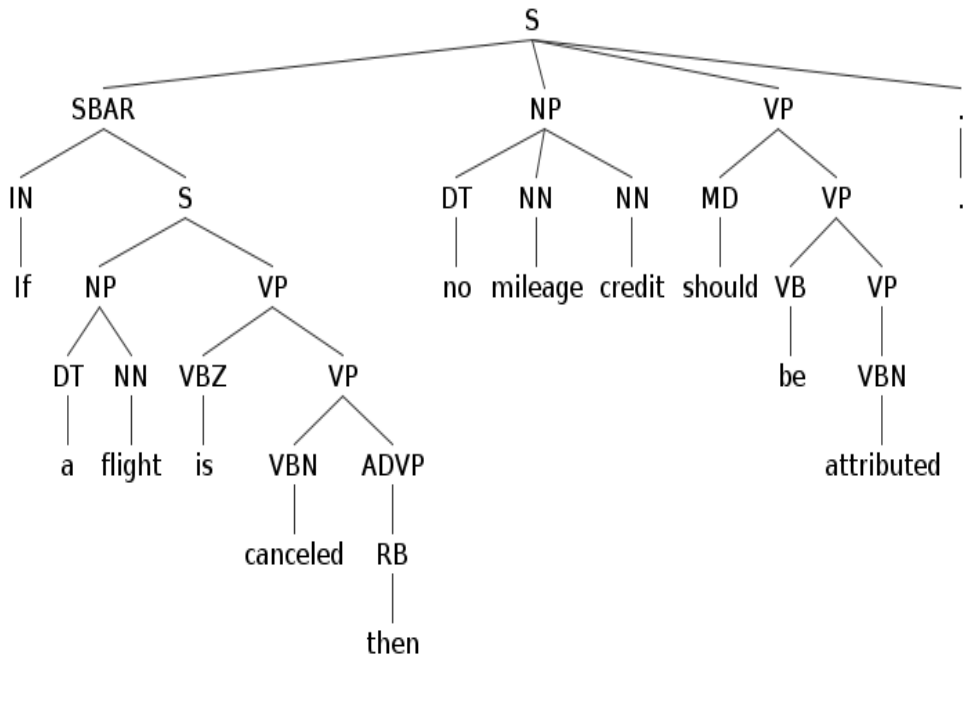



**Figure VI.23** – Exemple de l'analyse syntaxique d'une phrase avec Stanford parser.

```
<Sentence rdf:ID="sent0-n">
<content>If a flight is canceled, then no mileage credit should be attributed.</content>
<normalizedContent>If a flight is canceled then no mileage credit should be attributed</normalizedContent>
<parseTo rdf:ressource="#anaSsent0-n"/>
</Sentence>

<AnalyzedSentence rdf:ID="anaSsent0-n">
   <content>
        (ROOT
          (S
            (SBAR (IN If)
              (S
                (NP (DT a) (NN flight))
                (VP (VBZ is)
                  (VP (VBN canceled)
                    (ADVP (RB then))))))
            (NP (DT no) (NN mileage) (NN credit))
            (VP (MD should)
              (VP (VB be)
                (VP (VBN attributed))))
            (. .)))
   </content>
   <parser>Stanford Parser</parser>
</AnalyzedSentence>
```

**Figure VI.24** – Description d'une phrase dans le schéma.

| Types d'opérateurs | Valeurs |
|---|---|
| **Quantificateurs** | each, for, some, always, at least one, at least, at most one, at most, exactly one, exactly, at least, at most, more than one, never, can, can...only, need not, sometimes |
| **Logiques** | and, or, or...but not both, it is not the case that, if, then, if...then, if and only if, not both...and, neither...nor, whether or not |
| **Modaux** | shall, must, must not, may, it is obligatory that, it is prohibited that, it is necessary that, it is impossible that, it is possible that, it is permitted that, only if, may...only if, must not...if not, it is permitted that...only if, it is obligatory that not...if not, it is possible that...only if, it is necessary that not...if not |
| **Autres mots-clés** | the, a, an, another, a given, that, who, is of, what |

**Tableau VI.1** – Mots-clés tirés de la spécification SBVR.

### VI.5.1.2 Mots-clés

Les mots-clés caractéristiques de règles métier sont tirés des spécifications SBVR [7]. Ils correspondent, en anglais, aux opérateurs de quantification, logiques, modaux, etc. qui permettent la construction des formulations logiques de SBVR-SE. Dans le tableau VI.1, nous avons répertorié la plupart des mots-clés que nous avons trouvés. Ils sont composés de mots simples (un mot), de groupes de mots successifs ou de groupes de mots séparés (avec ...). Leur stockage est pris en compte dans le schéma de la structure d'index (VI.7) via l'objet "**KeyWord**", dont la propriété "**value**" contient la valeur textuelle. La figure VI.25 montre comment ils sont structurés dans le schéma : chaque objet "**KeyWord**" ne décrit qu'un seul mot, les groupes de mots successifs ou séparés sont pris en compte par la construction des patrons linguistiques exposée dans la section suivante.

```
<KeyWord rdf:ID="kw-01"><value>if</value></KeyWord>
<KeyWord rdf:ID="kw-02"><value>then</value></KeyWord>
<KeyWord rdf:ID="kw-03"><value>it</value></KeyWord>
<KeyWord rdf:ID="kw-04"><value>is</value></KeyWord>
<KeyWord rdf:ID="kw-05"><value>obligatory</value></KeyWord>
<KeyWord rdf:ID="kw-06"><value>that</value></KeyWord>
```

**Figure VI.25** – Représentation des mots-clés dans le graphe RDF.

7. http ://www.omg.org/spec/SBVR/1.0/

## VI.5.2 Implémentation des patrons linguistiques

Plusieurs types de patrons linguistiques ont été définis dans ce module selon le type de fouille à effectuer. Ils sont encapsulés dans des requêtes SPARQL qui sont ensuite appliquées sur le graphe RDF pour l'identification des phrases (objets "**Sentence**") qui matchent avec ces patrons.

Au fur et à mesure que les patrons sont appliqués et que des phrases de règles sont identifiées et validées par l'expert métier, la base documentaire est restreinte aux seules phrases non encore validées. Pour cela, nous utilisons les propriétés "**isValidated**" et "**isNoValidated**" de "**Sentence**" qui sont à **true** ou **false** selon que la phrase est validée comme règle métier ou pas. Ces propriétés sont en général modifiées par l'expert lui-même à partir de l'interface d'édition des règles candidates. Si elles sont toutes les deux à false, cela veut dire que la phrase n'a pas encore été étudiée par l'expert. Seules les phrases sont prises en compte dans les requêtes SPARQL.

### VI.5.2.1 Patrons pour l'analyse lexicale

Les patrons "lexicaux" s'appliquent sur les contenus bruts normalisés ("**normalizedContent**") des phrases de la base documentaire (la normalisation rend la recherche indépendante de la mise en page en remplaçant les suites de séparateurs par un blanc unique). Ils sont pris en compte dans une fonction de filtre du SPARQL appelé "**regex**" qui permet de définir des patrons sous forme d'expressions régulières [8]. Cette fonction, comme présenté sur la ligne suivante, renvoie un booléen et vérifie si un patron (simple literal pattern) matche avec une chaîne de caractères (simple literal text).

**xsd :boolean REGEX (simple literal text, simple literal pattern)**

Ces patrons permettent de vérifier si des mots-clés apparaissent ou non dans ces contenus. Ils dépendent des différentes types de combinaisons des mots-clés ($m_i$) possibles autour des opérateurs tels que "+", "**...**", "$\wedge$", "$\vee$". Nous avons alors plusieurs types de patrons $p$ notamment quand la recherche porte sur les phrases à partir :

1. D'un seul mot-clé $m$, $p$ = **"m"** (ex : $p$ = "must").
2. D'un groupe de mots-clés successifs $m_i$, $p$ = "$+_{i=1}^{n} m_i$" : dans la fonction regex cela donne $p$ = "$m_1$ $m_2$ $m_3$ ... $m_n$" (ex : $p$ = "at least one"), l'opérateur + est remplacé par un caractère blanc.

8. Les expressions régulières sont des motifs de caractères que l'on sait traduire automatiquement en automate de reconnaissance. La fonction regex est spécifiée dans http ://www.w3.org/TR/rdf-sparql-query/#funcex-regex

3. D'un groupe de mots-clés distincts $m_i$ qui apparaissent dans l'ordre mais éloignés les uns des autres, $p = ...{}_{i=1}^{n} m_i$ : dans la fonction regex cela donne $p = m_1.+m_2.+m_3.+....+m_n$, l'opérateur ... est remplacé par les caractères .+ pour dire qu'il y a au moins un caractère entre les différents mots-clés.
4. D'un groupe de mots-clés distincts soit qui apparaissent en même temps, $p = \wedge_{i=1}^{n} m_i$, soit dont au moins un apparaît, $p = \vee_{i=1}^{n} m_i$ : cela conduit alors à une intersection ou une union de filtres "**regex**" des trois types de patrons précédents ($p_1 \cap p_2 \cap ... \cap p_n$ ou $p_1 \cup p_2 \cup ... \cup p_n$ ) où chaque $p_i = m_i$ mais $m_i$ pourrait lui-même correspondre aux deuxième et troisième cas.

Des requêtes SPARQL sont alors implémentés pour chacun des types de patrons ci-dessus : les trois premiers types sont pris en charge par le schéma de requête VI.26 où seul $p$ varie, et le dernier utilise deux schémas de requêtes : (VI.27 et VI.28).

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?sentence ?content
where{
  ?sentence rdf:type schema:Sentence.
  ?sentence schema:content ?content.
  ?sentence schema:normalizedContent ?norm.

  ?sentence schema:isValidated <false>.
  ?sentence schema:isNoValidated <false>.

  FILTER regex(str(?norm), p)
}
```

**Figure VI.26** – Typologie de requêtes prenant en compte les mots-clés composé d'un mot, d'une succession de mots, et des mots éloignés les uns les autres.

### VI.5.2.2 Patrons pour l'analyse syntaxique

Les patrons "syntaxiques" constituent une extension des patrons "lexicaux", ils sont aussi construits autour des mots-clés mais utilisent des informations syntaxiques renvoyant aux catégories morpho-syntaxiques des mots d'une phrase et aux dépendances syntaxiques entre ces mots. Ces patrons sont donc appelés à s'appliquer sur les arbres syntaxiques des phrases de la base documentaire qui sont pris en charge dans les objets "**AnalyzedSentence**" avec la propriété "**content**" (héritée de "**TextualUnit**") qui stocke l'arbre syntaxique d'une phrase donnée.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?sentence ?content
where{
  ?sentence rdf:type schema:Sentence.
  ?sentence schema:content ?content.
  ?sentence schema:normalizedContent ?norm.

  ?sentence schema:isValidated <false>.
  ?sentence schema:isNoValidated <false>.

  FILTER regex(str(?norm), p1).
  FILTER regex(str(?norm), p2).
  ...
  FILTER regex(str(?norm), pn).
}
```

**Figure VI.27** – Typologie de requêtes prenant en compte les combinaisons de mots-clés à base d'intersections de patrons.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?sentence ?content
where{
  ?sentence rdf:type schema:Sentence.
  ?sentence schema:content ?content.
  ?sentence schema:normalizedContent ?norm.

  ?sentence schema:isValidated <false>.
  ?sentence schema:isNoValidated <false>.

  {FILTER regex(str(?norm), p1).}
  UNION
  {FILTER regex(str(?norm), p2).}
  UNION
  ...
  UNION
  {FILTER regex(str(?norm), pn).}
}
```

**Figure VI.28** – Typologie de requêtes prenant en compte les combinaisons de mots-clés à base d'union de patrons.

Ces patrons ne sont pas implémentés dans des requêtes SPARQL, mais utilisent une requête pour l'accès aux phrases auxquelles ils vont s'appliquer. La requête VI.29 fournit l'ensemble des

arbres syntaxiques associés aux phrases de la base documentaire non encore classifiées. Chaque patron construit est comparé avec chacun de ces arbres et les phrases filtrées sont proposées à l'expert métier pour les valider comme règles candidates.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?sentence ?content ?analyzedContent
where{
  ?sentence rdf:type schema:Sentence.
  ?sentence schema:content ?content.
  ?sentence schema:parsedTo ?analyzedSentence.
  ?analyzedSentence schema:content ?analyzedContent.

  ?sentence schema:isValidated <false>.
  ?sentence schema:isNoValidated <false>.

}
```

**Figure VI.29** – Requête renvoyant la liste des phrases analysées syntaxiquement dans la base documentaire.

Les patrons syntaxiques sont construits à partir de mots-clés $m_i$ et d'informations syntaxiques $c_j$. Ils utilisent des opérateurs tels que "+" et "**...**" et les relations hiérarchiques des éléments $m_i$ et $c_j$ dans l'arbre syntaxique. Nous avons défini deux catégories de patrons, l'une sur les arbres syntaxiques des composants ($P_{asc}$), l'autre sur les arbres syntaxiques des dépendances ($P_{asd}$).

Nous avons testé l'implémentation dans ce module des patrons sur les arbres de composants. A la différence des patrons définis au niveau lexical, SPARQL n'est pas adapté pour implémenter les patrons devant s'appliquer sur des arbres syntaxiques : d'une part la structure des arbres syntaxiques est plus complexe que celle des chaînes de caractères, d'autre part les catégories syntaxiques ne sont pas prises en charge dans les expressions régulières classiques qui traitent des caractères. Pour cette raison, nous avons testé l'outil Tregex [9] proposé par le projet Stanford [10], le même qui propose aussi l'analyseur syntaxique que nous utilisons.

Trois types de patrons $p$ peuvent être représentés dans Tregex et appliqués sur les arbres de composants permettant une recherche sur les phrases à partir d'une combinaison de mots-clés et de catégories syntaxiques. :

1. L'opérateur "+", d'où les patrons $p = +_{i=1}^{n}(m_i|c_i) \wedge_{j=1}^{k} ancestor(c_i, c_j)$ qui sont associés

9. http ://nlp.stanford.edu/software/tregex.shtml
10. http ://nlp.stanford.edu/software/index.shtml

aux mots-clés successifs du genre $+_{i=1}^{m} m_i$. Dans Tregex, dans ce même patron la combinaison est encapsulée dans les liens de succession entre les $c_i$ et les $c_j$. Pour généraliser par exemple l'arbre syntaxique de la phrase dans la figure VI.31, nous pouvons définir un patron $P_1$ (voir VI.30) de recherche du mot-clé "**must**" dans la base documentaire analysée syntaxiquement.

**$P_1$ : (MD < < must) > ((VP $. NP) <- (VP <, VB))**
**$P_2$ : (MD < < must) > ((VP $. NP) <- (VP < VB < < be))**
**$P_3$ : If > > IN > (SBAR $ (ADVP < RB < < then))**

**Figure VI.30** – Exemple de patrons de fouille syntaxique applicables sur des arbres syntaxiques des composants.

Ce patron $P_1$ filtre toutes les phrases où :
- le mot-clé "must" est dominé par le nœud MD (opérateur modal), à son tour immédiatement dominé par un nœud VP (groupe verbal)
- le frère direct à gauche de ce VP est un nœud NP (groupe nominal)
- le dernier fils à droite de ce VP est un autre nœud VP dont le premier fils est un verbe VB.

Autrement dit, ce patron reconnaît toutes les phrases où le mot-clé "must" est précédé d'un groupe nominal NP et suivi d'un groupe verbal VP. Ce patron peut être plus spécialisé en précisant le verbe VB du groupe verbal VP. D'où le patron $P_2$ qui est une spécialisation de $P_1$ où le nœud VB domine "**be**".

2. L'opérateur "**...**", d'où les patrons $p = ..._{i=1}^{n}(m_i|c_i) \wedge_{j=1}^{k} ancestor(c_i, c_j)$ qui sont associés aux mots-clés successifs mais éloignés les uns les autres. Dans Tregex, pour ce même patron la combinaison est encapsulée dans les liens de succession entre les $c_i$ et les $c_j$. Par exemple pour la phrase VI.32, nous pouvons avoir un patron $P_3$ (voir VI.30) qui matche avec toutes les phrases où apparaît le mot-clé "**if...then**". Ce patron $P_3$ permet de trouver les phrases commençants par "**If**" dominé par le nœud SBAR (sous-phrase) qui est frère d'un nœud ADVP (adverbe pronominal) dominant une chaîne avec le nœud RB puis la valeur "**then**".

3. La combinaison de ces deux premiers opérateurs, "+" et "**...**" qui permet de construire des patrons autour de mots-clés plus longs et plus complexes comme "**must not...if not**", d'où des patrons dans Tregex décrits par $p = (+|...)_{i=1}^{n} \wedge_{j=1}^{k} ancestor(c_i, c_j)(m_i|c_i)$. Ces patrons sont définis de la manière que les premiers types mais semblent plus rares ; nous

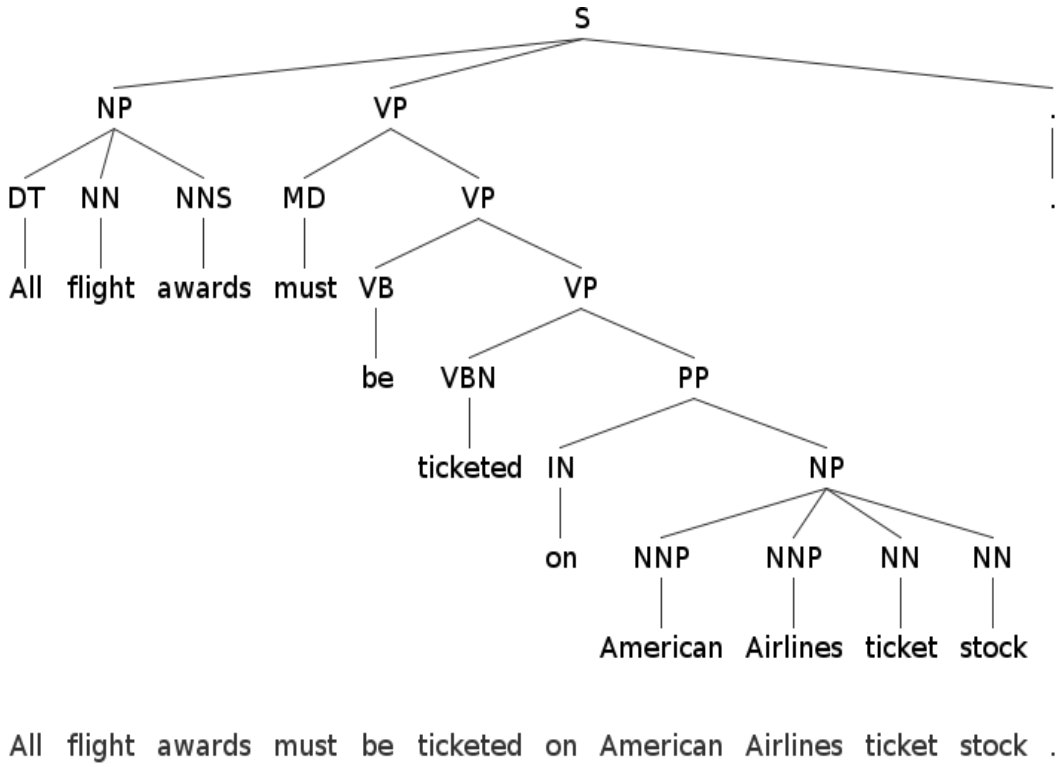



**Figure VI.31** – Arbre syntaxique des composants d'une phrase à partir duquel est défini le patron syntaxique.

n'avons pas trouvé d'exemple dans nos expérimentations mais ils pourront jouer un rôle dans d'autres corpus.

L'utilisation de ces patrons bénéficierait grandement de la constitution d'une bibliothèque de patrons avec l'aide de spécialistes de la syntaxe. Dans la mesure où une bonne partie de la détection repose sur les mots-clés, cette bibliothèque aurait une bonne réutilisabilité. Nous avons pour l'instant montré comment intégrer ces patrons à SemEx et comment les utiliser pour des fouilles plus complexes.

#### VI.5.2.3 Patrons pour l'analyse sémantique

Les patrons construits dans la fouille sémantique reposent sur les concepts ontologiques, et les termes métier qu'ils décrivent, qui ont servis à annoter des règles métier déjà produites et stockées dans la base de règles afin d'identifier les phrases annotées par ces mêmes concepts. Comme nous l'avons dit dans le chapitre précédent, cette fouille sert à compléter la fouille syntaxique et aide à retrouver les règles candidates enfouies dans les textes et qui ne sont pas décrites par des mots-clés. Nous avons ainsi deux types d'implémentations possibles pour

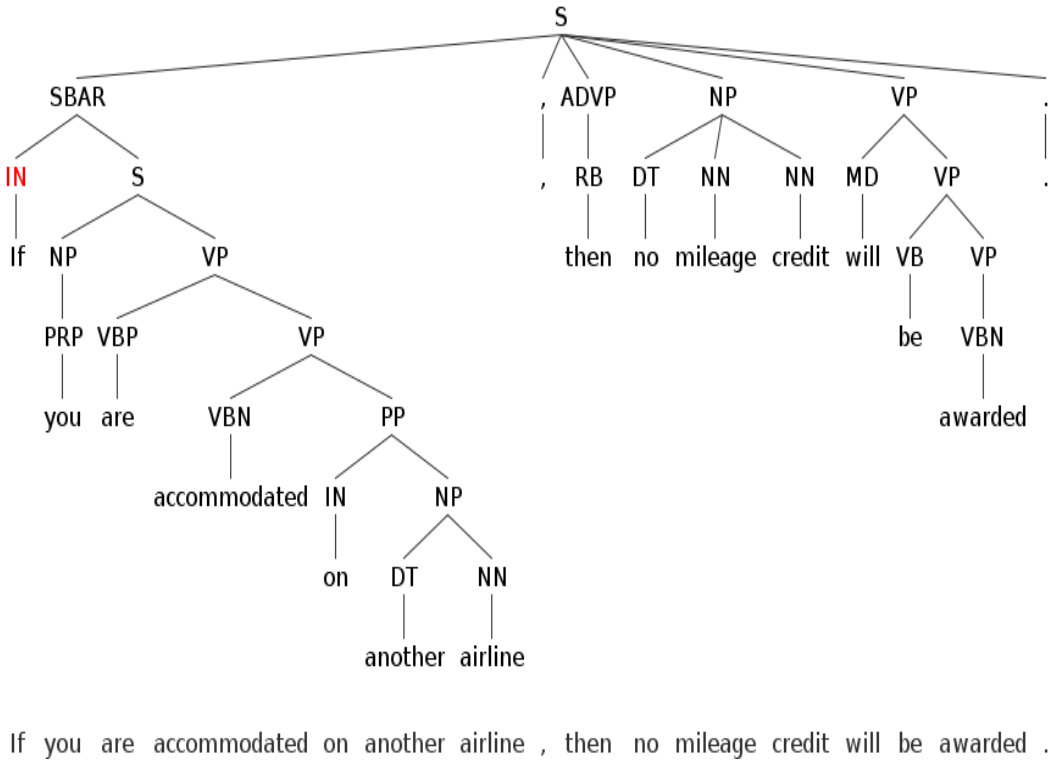



**Figure VI.32** – Arbre syntaxique des composants d'une phrase à partir duquel est défini le patron syntaxique.

mettre en œuvre cette fouille conceptuelle :

- La première implémentation consiste à considérer les concepts les plus pertinents pour décrire les règles candidates déjà extraites, ceux qui sont le plus fréquemment utilisés pour annoter les règles candidates, et à chercher s'ils ont servi à annoter des phrases non encore validées. Il s'agit alors d'un processus en deux phases, une première pour déterminer ces concepts dans la base de règles et une seconde pour sélectionner les phrases annotées par ces derniers. Ces deux phases sont implémentées avec deux requêtes SPARQL (voir figure VI.33) à appliquer sur la structure d'index via le module d'accès aux données et qui sont utilisées l'une après l'autre. La requête (1) renvoie la liste de tous les éléments ontologiques (concepts, relations et instances) annotant les règles candidates de la base de règles. Le résultat de cette requête est présenté via l'interface de navigation, l'expert peut alors sélectionner le concept qu'il souhaite utiliser, l'idée est qu'il choisisse parmi ceux dont la fréquence est plus élevée. La requête (2) associée à la phase 2 est définie à partir de ce concept.
- La seconde consiste à rechercher de façon plus simple toutes les phrases non encore validées et qui partagent au moins un concept (respectivement relation ou instance) avec une règle

```
(1) : PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?ontologicalElement, (count(distinct ?ontologicalElement) as ?frequence)
where
{
  ?rule rdf:type schema:CandidateRule.
  ?rule schema:annotatedBy ?textlink.

  {?textlink schema:defineConcept ?ontologicalElement.}
  UNION
  {?textlink schema:defineRelation ?ontologicalElement.}
  UNION
  {?textlink schema:defineInstance ?ontologicalElement.}

}
ORDER BY ?ontologicalElement DESC(?frequence)


(2) : PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX onto: <http://lipn.univ-paris13.fr/RCLN/terminae/xxx#>
select ?sentence, ?content
where
{
  ?sentence rdf:type schema:Sentence.
  ?sentence schema:annotatedBy ?textlink.
  ?sentence schema:content ?content.

  {?textlink schema:defineConcept <onto:EO>.}
  UNION
  {?textlink schema:defineRelation <onto:EO>.}
  UNION
  {?textlink schema:defineInstance <onto:EO>.}

  ?sentence schema:isValidated <false>.
  ?sentence schema:isNoValidated <false>.

}
```

**Figure VI.33** – La requête (1) renvoie la fréquence des concepts qui annotent les règles candidates de la base de règles dans l'ordre décroissant.
La requête (2) renvoie les phrases annotées par un élément ontologique "**EO**" donné.

candidate donnée. Ici aussi, de la même manière que dans la première fouille sémantique, nous avons défini une requête SPARQL permettant de déterminer toutes ces phrases à

partir d'une règle candidate qu'aura choisie l'expert métier. La requête VI.34 identifie d'abord tous les éléments ontologiques annotateurs d'une règle candidate donnée ("**rc**") et ensuite toutes les phrases annotées par ces derniers.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select distinct ?sentence ?content.
where
{
  <rc> schema:annotatedBy ?textlink.

  {?textlink schema:defineConcept ?ontologicalElement.}
  UNION
  {?textlink schema:defineRelation ?ontologicalElement.}
  UNION
  {?textlink schema:defineInstance ?ontologicalElement.}

  ?sentence rdf:type schema:Sentence.
  ?sentence schema:annotatedBy ?link.
  ?sentence schema:content ?content.

  ?link schema:defineResource ?ontologicalElement.

  ?sentence schema:isValidated <false>.
  ?sentence schema:isNoValidated <false>.

}
```

**Figure VI.34** – Cette requête renvoie la liste de toutes les phrases qui partagent les mêmes concepts qu'une règle candidate "**rc**" donnée.

## VI.6 Module de Normalisation

Ce module est interconnecté à l'interface d'édition des règles candidates et implémente les opérations de normalisation. Il interagit aussi avec le module MAD afin d'accéder aux ressources qu'il manipule. Son but est d'aider à l'automatisation de ces opérations tout en favorisant l'interactivité avec l'expert. Nous avons la possibilité d'attacher à l'interface d'édition, pour chaque opération de normalisation des règles candidates, des actions d'aide à l'exécution de l'opération. Elles sont lancées par l'expert et lui signalent les phrases où cette opération pourrait

s'appliquer ou même lui proposent une reformulation. Pour achever l'intégration des outils et techniques nécessaires à chaque type de normalisation, la présentation à l'expert de leurs résultats nécessite un travail méthodologique qui reste largement à faire. Des expérimentations sur les outils syntaxiques non encore intégrés sont rapportées dans le chapitre VII.

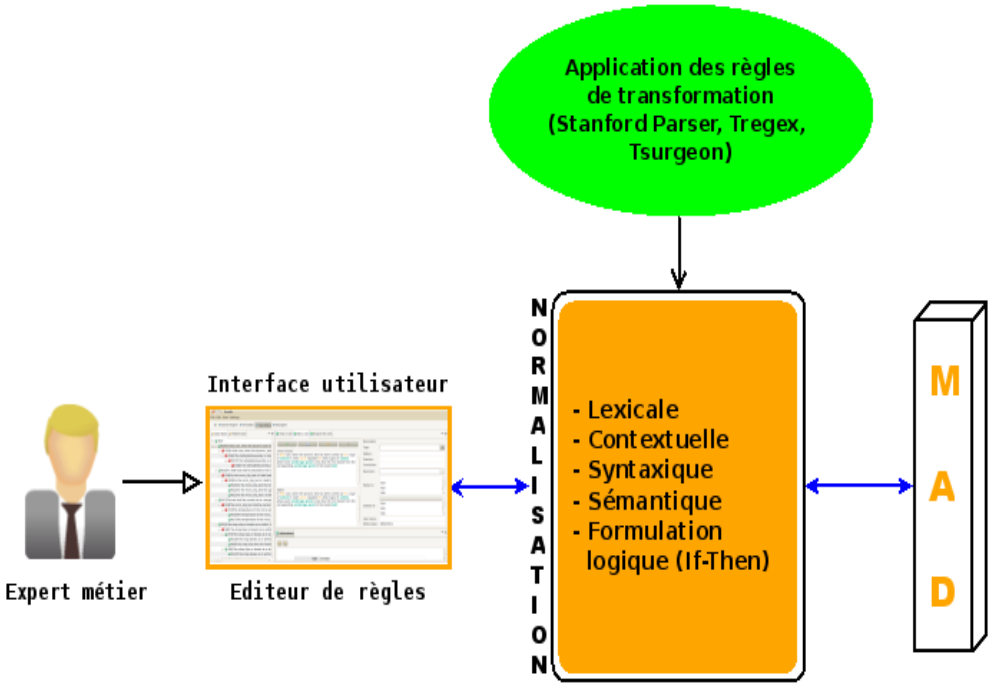



**Figure VI.35** – Module de normalisation des règles candidates.

## VI.6.1 Implémentation des opérations de normalisation

Nous décrivons ici les outils et techniques que nous avons intégrés dans SemEx ou qui pourront être intégrés pour faciliter les opérations de normalisation.

### VI.6.1.1 Niveau lexical

Nous avons vu que la normalisation lexicale (voir V.3.1.1) consiste à stabiliser le vocabulaire métier utilisé dans les règles candidates conformément à l'ontologie descriptive du vocabulaire de domaine. L'idée est d'utiliser les termes métier qui désambiguïsent au mieux le vocabulaire utilisé dans une règle candidate donnée. Comme chaque élément conceptuel (concepts, instances, relations) est associé à une liste de termes métier pouvant le désigner, et que l'un est choisi comme terme préférentiel, il s'agit d'utiliser les termes métier les plus précis dans une

règle candidate. Pour cela, nous considérons les annotations des règles candidates au regard de l'ontologie et nous proposons dans SemEx deux façons de procéder :

- Une façon manuelle où l'expert choisit lui-même les termes métier qu'il juge plus précis. Ce travail est effectué à partir de l'interface d'édition qui lui propose la liste des termes métier préférentiels associés à un élément conceptuel. Ce choix est mis en œuvre à travers un système d'auto-complétion. Ici, l'expert choisit l'emplacement de l'opération (insertion ou remplacement) et la normalisation consiste à contrôler le vocabulaire ajouté.
- Une façon automatique où nous implémentons un algorithme permettant de remplacer tous les termes métier utilisés par les termes préférentiels associés à leurs concepts (instances ou relations) annotateurs. Cet algorithme permet de proposer à l'expert une normalisation lexicale complète qu'il peut accepter ou refuser. Il est intégré au module de normalisation, interroge le module MAD et ses résultats peuvent être visualisés directement sur l'interface d'édition.

Dans les deux cas, il est nécessaire d'accéder à la liste des termes métier synonymes à un terme métier qui est annoté par un élément conceptuel donné. Ces informations sont disponibles à travers le module MAD dans lequel nous avons défini une requête SPARQL (voir VI.36) qui s'applique sur la structure d'index.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
select ?ts
where
{
  ?ec rdf:type schema:Concept.
  ?ec schema:realizeConcept <un-concept-donne>.
  {?ec skos:prefLabel ?ts.} UNION {?ec skos:altLabel ?ts.}
}
```

**Figure VI.36** – Cette requête retourne la liste des termes métier synonymes ?*ts* (terme préférentiel en tête) qui peuvent être annotés par un concept donné "*un* − *concept* − *donne*". Nous utilisons Instance et realizeInstance pour les instances, et Relation et realizeRelation pour les relations.

L'algorithme de normalisation lexicale (voir Algorithme 1) fait aussi intervenir des requêtes SPARQL. Il prend en entrée la règle candidate à normaliser ainsi que l'ensemble des termes annotés de la règle candidate. Pour chacun de ces termes, nous identifions l'élément ontologique attaché (voir VI.37) et la liste des termes synonymes associés à celui-ci (voir VI.36).Tout terme métier différent du terme préférentiel de "**elOntAnnotateur**" est remplacé par ce terme dans le contenu textuel de la règle candidate.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?p ?q ?ec
where
{
  <RC> rdf:type schema:CandidateRule.
  <RC> schema:annotatedBy ?textlink.
  ?textlink schema:startOffset <p>.
  ?textlink schema:endOffset <q>.
  ?textlink schema:defineResource ?ec.
  ?ec rdf:type schema:OntologyResource.
}
```

**Figure VI.37** – Cette requête retourne la liste des termes métier annotée dans une "**RC**" donnée, notamment leurs positions ($?p$ et $?q$) dans celle-ci, ainsi que leurs éléments ontologiques (concepts, relations ou instances) annotateurs.

#### VI.6.1.2 Niveau contextuel

Nous avons vu que la normalisation contextuelle (voir V.3.1.2) consiste à remplacer les mots ou symboles qui ont besoin du contexte pour être interprétés par les termes métier auxquels ils renvoient. Il s'agit notamment des mots grammaticaux, des clés de référence, des relations cachées ainsi que des termes métier génériques. Le rôle du module de normalisation est alors d'aider l'expert à la fois dans l'identification du contexte d'une règle candidate dans son texte d'origine mais aussi dans le remplacement de ces mots par les termes métier exacts. Chacune des ces deux tâches peut être effectuée de façon entièrement manuelle par l'expert métier mais le module lui propose des requêtes SPARQL applicables sur le graphe RDF de la structure d'index pour déterminer le contexte d'une règle candidate. On peut aussi envisager des outils de résolution d'anaphores pour rétablir le contexte. Toutefois, dans cette thèse, nous avons mis l'accent sur la conception de la plateforme et la méthodologie d'acquisition qu'elle permet de mettre en œuvre plutôt que sur l'étude de l'impact précis des outils de TAL et leur intégration effective.

Le contexte d'une règle candidate correspond à l'ensemble des phrases nécessaires à sa compréhension. Ces phrases sont généralement sa version originale, celles qui la précédent et celles qui la suivent. Mais il peut aussi s'agir de phrases éloignées dans le texte (par exemple quand deux phrases sont dans des paragraphes différents). La requête SPARQL (VI.38) répond au premier cas, à l'aide des relations "**before**" et "**after**" du schéma formel de la structure d'index. Des variantes de cette requête permettent d'obtenir un nombre donné de phrases apparaissant avant et après la règle candidate dans le texte.

**Procédure** Normalisation lexicale(RC : **Règle Candidate**, TermesAnnotés : **Tableau[N]**)
    i : **entier** ;
    **Pour** (i←0 ; i<N ; i++) **faire**
        tm ← TermesAnnotés[i] ;//*obtenu avec VI.37*
        elOntAnnotateur = annotéPar (tm) ;//*l'élément ontologique annotateur obtenu avec VI.37*
        T = liste-termes (elOntAnnotateur) ;//$T = t_p, t_{s1}, t_{s2}, ..., t_{sm}$ *: liste des termes métier synonymes associés à l' "elOntAnnotateur" obtenue avec VI.36.*
        **Si** (tm <> tp) **Alors**
            remplacer(RC,tm,tp) ; //*Remplacer tm par tp dans la règle RC*
        **Fin Si**
    **Fin Pour**
**Fin**

**Algorithme 1** – Algorithme de normalisation lexicale.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select DISTINCT ?s1 ?s ?s2
where
{
  ?s1 rdf:type schema:Sentence.
  ?s2 rdf:type schema:Sentence.

  <RC> rdf:type schema:CandidateRule.
  ?s rdf:type schema:Sentence.
  ?s schema:annotatedBy ?textlink.
  ?textlink schema:defineRule <RC>.

  optional{
    ?s1 schema:before ?s.
  }
  optional{
    ?s2 schema:after ?s.
  }
}
```

**Figure VI.38** – Requête retournant les deux phrases qui entourent une règle candidate "RC" donnée.

Ainsi, actuellement dans SemEx, pour chacun des cas de contextualisation (mots grammaticaux, clés de référence, relations cachées ou termes métier génériques), il appartient à l'expert d' utiliser la requête la plus adaptée pour la normalisation d'un cas donné. Si les phrases qui entourent de très près une règle candidate ne suffisent pour faire ce travail, il faut agrandir la fenêtre ou utiliser d'autres critères de structure (titres, items) afin de visualiser les phrases les plus éloignées dans les textes.

#### VI.6.1.3 Niveau syntaxique

Les règles candidates subissent aussi une normalisation au niveau de leur structure syntaxique, soit pour la réordonner ou pour la simplifier, soit pour la découper en sous règles candidates (indépendantes). Il s'agit d'une tâche très délicate qui peut facilement modifier la valeur sémantique d'une règle candidate. SemEx propose à l'expert une interface d'édition qui permet de faire ou de contrôler le travail. Nous avons testé dans ce module des outils pour accompagner le travail de l'expert. Nous nous sommes inspirés des travaux de simplification textuelle qui ont utilisé des patrons de transformation. Ceux-ci permettent de proposer à l'expert une normalisation syntaxique calculée automatiquement qu'il peut valider via l'interface d'édition.

Nous nous appuyons sur les patrons définis dans [Chandrasekar et al., 1996] et dans [Chandrasekar and Srinivas, 1997]. Nous n'avons pas adapté à notre contexte interactif l'acquisition automatique qu'ils ont expérimentée et nous nous sommes limité à tester une définition manuelle des ces patrons.

Les patrons de normalisation sont définis et appliqués sur des règles candidates analysées syntaxiquement. Techniquement nous utilisons l'analyseur syntaxique Stanford Parser, parce que nous l'avons testé dans le module d'identification des règles candidates et que nous disposons d'outils TAL, notamment de Tregex et Tsurgeon qui sont issus du projet de Stanford [11], pour traiter la sortie de cet analyseur. Tregex reconnaît des nœuds conformes à une expression régulière d'arbre, et Tsurgeon transforme un arbre en modifiant un schéma donné. Les patrons sont alors définis manuellement à partir des arbres syntaxiques de règles candidates obtenus avec le Stanford Parser. Ils sont constitués de deux parties, une prémisse qui spécifie sur quel nœud de l'arbre syntaxique une modification doit être effectuée, et une conclusion qui spécifie l'opération de normalisation que doit subir cet arbre. Ainsi Tregex permet de retrouver la prémisse sur l'arbre syntaxique de la règle candidate et Tsurgeon applique l'opération sur ce même arbre pour modifier la règle candidate.

11. http ://nlp.stanford.edu/software/

Tous ces outils sont implémentés en Java ce qui facilite leur intégration dans SemEx.

En pratique, nous avons mis tout cela en œuvre dans le module de normalisation suivant un procédé en plusieurs étapes que nous décrivons ci-dessous :

(i) Sélection d'exemples significatifs correspondant à un ensemble de couples de règles candidates complexes (non normalisées et nécessitant une normalisation) et normalisées (comme les exemples cités dans V.3.1.3). Ces exemples tests sont produits manuellement dans le module et proviennent des textes réglementaires qui sont nos cas d'usage, mais d'autres exemples tirés d'autres sources ne feraient qu'enrichir la méthode.
Considérons le couple (e, e') tiré du corpus Aadvandage :

(e) Upgrades are void if sold for cash or other consideration.
(e') **If sold for cash or other consideration then upgrades are void**.

Nous montrons ci-après le résultat de l'analyse syntaxique de (e) et la manière dont un patron de normalisation est construit. Dans ce couple, le normalisation syntaxique réalise un réordonnancement de la structure d'une règle candidate ; plus précisément, au patron **B If A** est substitué **If A then B**.

(ii) Analyse syntaxique des couples d'exemples avec Stanford Parser. Chaque règle candidate complexe est analysée ainsi que l'ensemble de ses règles candidates normalisées. Cela permet de produire un arbre syntaxique pour chaque règle candidate analysée. L'analyse syntaxique du couple donné en exemple est représentée dans les figures VI.39 et VI.40.

(iii) Définition manuelle des patrons de normalisation à partir des couples d'arbres syntaxiques issus des couples (règle candidate complexe - règle candidate normalisée). Nous pouvons définir un patron de normalisation pour chaque couple qui traduit le passage d'une règle candidate complexe vers une règle candidate normalisée. La prémisse est définie à partir d'un arbre syntaxique et la conclusion du patron est à appliquer aussi sur un arbre syntaxique. Le plus souvent, il y a besoin de plusieurs transformations successives pour un seul patron de normalisation syntaxique d'une règle candidate. Dans la figure VI.41, nous avons défini deux transformations pour transformer $e$ en $e'$. Elles constituent le patron **p**. La première transformation **t1** permet de déplacer après la conjonction "**If**" la partie de l'arbre qui vient avant celle-ci. La prémisse de **t1** (voir figure VI.42) permet de pointer sur le nœud à déplacer : il s'agit du nœud **SBAR**, supportant la conjonction "**If**" via un nœud fils **IN**, avec la spécification qu'il est placé sous les nœuds successifs **ADJP**, **VP**, **S**. La conclusion de t1 permet de déplacer ce nœud pour l'installer comme le premier des nœuds fils de **S**. Vient ensuite la transformation t2 (voir figure VI.43). La prémisse de

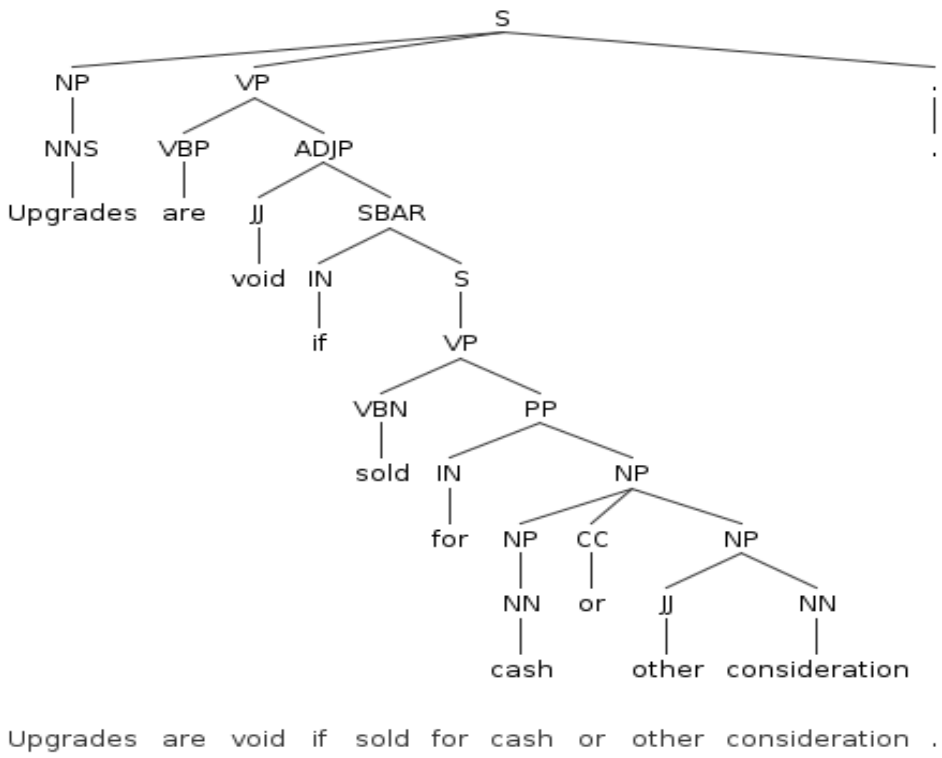



**Figure VI.39** – Analyse syntaxique de e.

**t2** permet de pointer sur le nœud **VP** surmonté des nœuds **S** et **SBAR** et sa conclusion permet d'ajouter la chaîne de trois nœuds **ADVP**, **RB** et "**then**" en l'insérant comme dernier fils du nœud **VP**.

(iv) Application des patrons sur d'autres règles candidates. Sur la figure VI.44, nous avons une règle candidate dont la structure ressemble à celle de *e* et donc il est possible de lui appliquer le patron **p** pour la normaliser syntaxiquement. Une application des deux patrons permet de déplacer le nœud caractérisant le bloc **if** et ensuite d'ajouter un nœud **then**, comme décrit sur la figure VI.45.

## VI.7 Module de maintenance

Ce module prend en charge le diagnostic des problèmes de maintenance identifiés précédemment (voir V.4). Il s'agit essentiellement d'exploiter la traçabilité des règles candidates tout au long de leur processus d'acquisition à partir des textes d'origine. En pratique, dans SemEx, c'est l'expert qui détecte lui-même un problème, ensuite via une l'interface de requête il fait appel à ce module pour faire le diagnostic du problème à partir duquel il pourra décider de

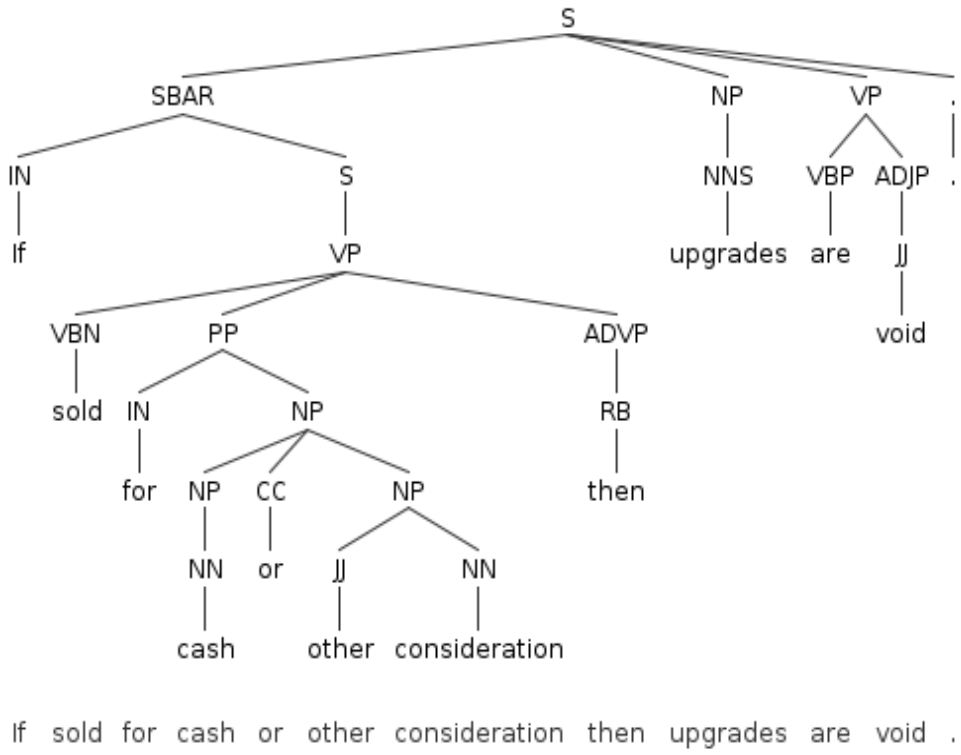



**Figure VI.40** – Analyse syntaxique de e'.

(**t1**) :
***Prémisse*** : (SBAR=sbar < IN < < If) > (ADJP > (VP > S=s))
***Conclusion*** : move sbar >1 s

(**t2**) :
***Prémisse*** : VP=vp > (S > SBAR)
***Conclusion*** : insert (ADVP (RB then)) >‘ vp

**Figure VI.41** – Les transformations t1 et t2 de normalisation de e en e'.

mettre à jour les règles affectées dans l'interface d'édition des règles.

Pour répondre aux divers problèmes de maintenance, nous avons implémenté dans ce module un ensemble de requêtes SPARQL destinées à aider l'expert dans ce travail de diagnostic :

- une requête (Figure VI.47) pour déterminer l'historique des reformulations subies par une règle candidate dans la phase de normalisation. Celle-ci permet de diagnostiquer un problème d'incomplétude dénotant des erreurs à certaines étapes de la normalisation. Elle permet aussi de diagnostiquer l'obsolescence d'une règle candidate déjà existante ;
- une requête (Figure VI.48) pour déterminer les phrases d'origine de deux règles candidates données. Celle-ci permet de trouver l'origine d'une incohérence (exception, redondance,

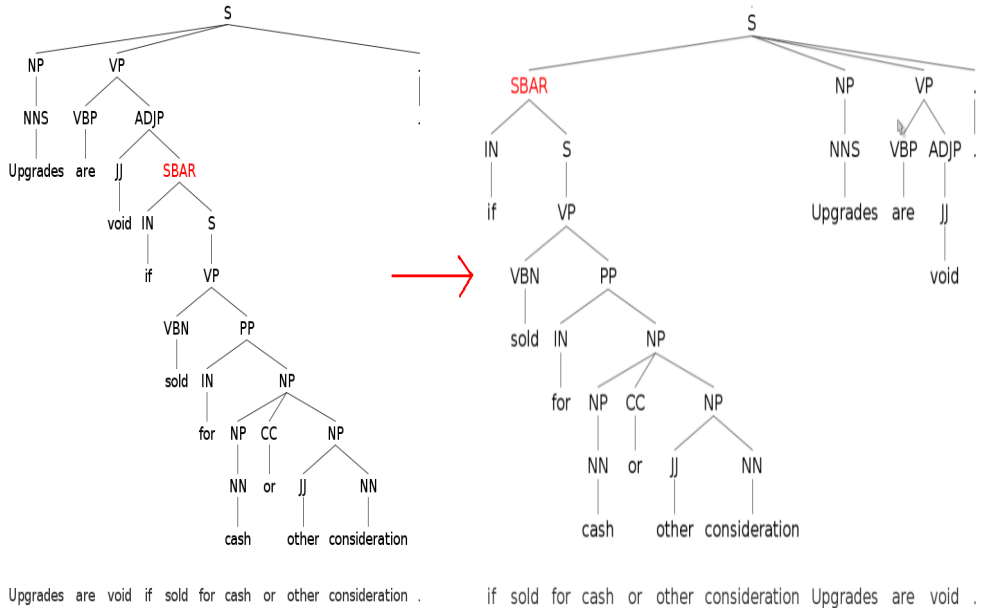



**Figure VI.42** – Application du patron p sur la règle candidate $e$ : transformation t1.

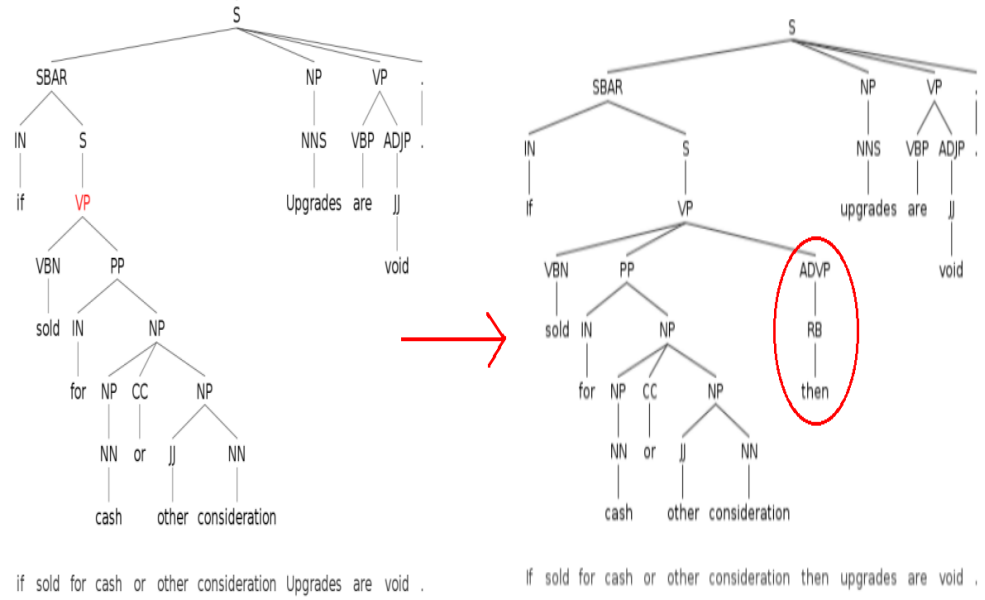



**Figure VI.43** – Application du patron p sur la règle candidate $e$ : transformation t2.

incompatibilité) entre deux règles candidates ;

– une requête (Figure VI.49) pour déterminer le contexte de la phrase d'origine d'une règle candidate donnée. Celle-ci est utilisée pour diagnostiquer une incomplétude, notamment lorsque les différentes version de reformulation ne suffisent pour le faire. Cette requête est similaire à celle utilisée dans le module de normalisation, au niveau contextuel ;

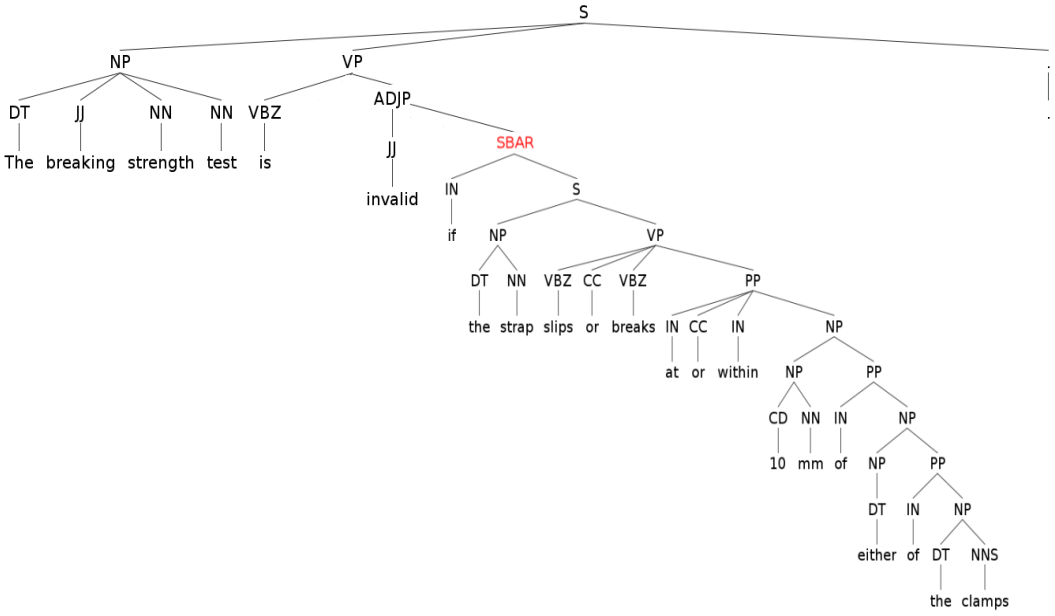



**Figure VI.44** – Application du patron p sur une nouvelle règle candidate.

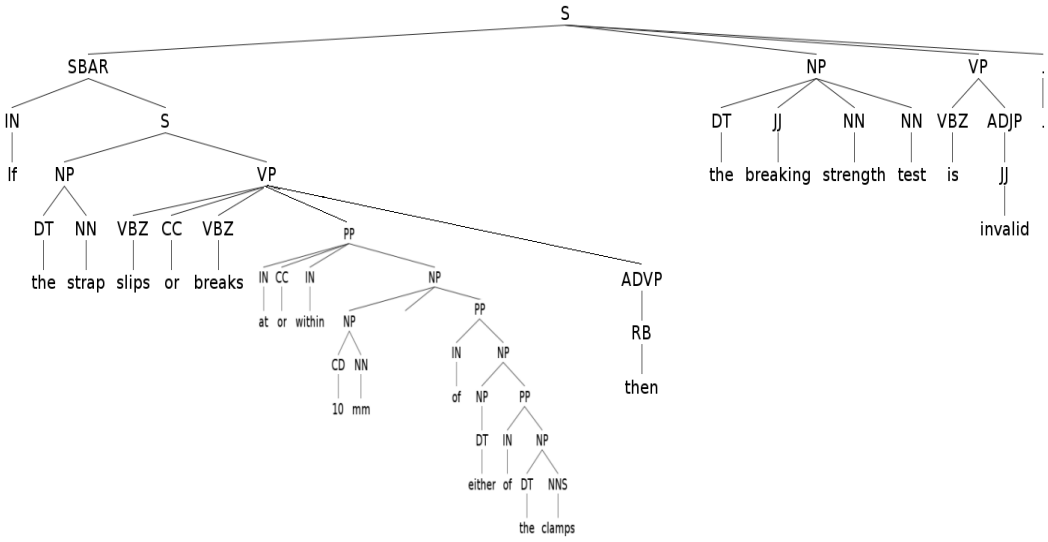



**Figure VI.45** – Application du patron p sur une nouvelle règle candidate : résultat.

– une requête (Figure VI.50) pour déterminer toutes les règles candidates dérivées d'une phrase donnée. Celle-ci permet un contrôle quand on soupçonne l'obsolescence de la phrase

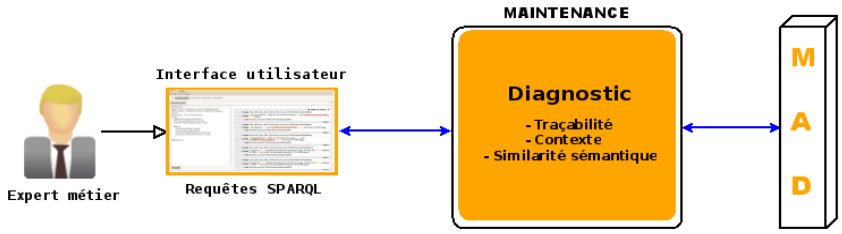



**Figure VI.46** – Module de maintenance des règles candidates.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?candidateRule, ?content
where
{
<rcID> rdf:type schema:CandidateRule.

{?candidateRule schema:previousForm <rcID>.}
UNION
{<rcID> schema:subRule ?candidateRule.}

?candidateRule schema:content ?content.

}
GROUP By DESC (?candidateRule)
```

**Figure VI.47** – Cette requête renvoie toutes les versions de reformulation d'une règle candidate "rcID" (son identifiant) donnée dans l'ordre descendant des identifiants des règles candidates trouvées.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select distinct ?sentence1, ?content1, ?sentence2, ?content2
where
{
<rcID-1> rdf:type schema:CandidateRule.
<rcID-1> schema:originalRuleVersion ?sentence1.
?sentence1 schema:content ?content1.

<rcID-2> rdf:type schema:CandidateRule.
<rcID-2> schema:originalRuleVersion ?sentence1.
?sentence1 schema:content ?content1.


}
```

**Figure VI.48** – Requête renvoyant le couple de phrases d'origine à partir d'un couple de règles candidates.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select DISTINCT ?s1 ?s ?s2
where
{
?s1 rdf:type schema:Sentence.
?s2 rdf:type schema:Sentence.

?s rdf:type schema:Sentence.
?s schema:annotatedBy ?textlink.
?textlink schema:defineRule <rcID>.

{?s1 schema:before ?s.}UNION {?s2 schema:after ?s.}

}
```

**Figure VI.49** – Cette requête renvoie le contexte de la phrase d'origine d'une règle candidate "rcID" (son identifiant) donnée.

dans le texte d'origine ;

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select distinct ?candidateRule, ?content
where
{
<sentenceID> rdf:type schema:Sentence.
<sentenceID> schema:annotatedBy ?textlink.

?textlink schema:defineRule ?candidateRule.

filter not exists{?candidateRule schema:previousForm ?fils1.}
filter not exists{?fils2 schema:subRule ?candidateRule.}

?candidateRule schema:content ?content.


}
```

**Figure VI.50** – Cette requête renvoie toutes les règles dérivées d'une phrase "sentenceID" (son identifiant) donnée.

– une requête (Figure VI.51) pour déterminer toutes les règles candidates ayant une similarité sémantique, i.e. qui partagent des concepts (respectivement des relations ou instances). Celle-ci peut contribuer à identifier les règles candidates de la base de règles qui sont susceptibles d'interagir avec une règle candidate donnée. La sélection se fait

uniquement à partir des éléments ontologiques qu'elles partagent.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?candidateRule, ?content
where
{
<rcID> rdf:type schema:CandidateRule.
<rcID> schema:annotatedBy ?textlink.

{?textlink schema:defineConcept ?ontologicalElement.}
UNION
{?textlink schema:defineRelation ?ontologicalElement.}
UNION
{?textlink schema:defineInstance ?ontologicalElement.}

?candidateRule rdf:type schema:CandidateRule.
?candidateRule schema:annotatedBy ?link.
?candidateRule schema:content ?content.

?link schema:defineResource ?ontologicalElement.

}
```

**Figure VI.51** – Cette requête renvoie la liste des règles candidates qui partagent au moins un élément ontologique avec une règle candidate "rcID" (son identifiant) donnée.

## VI.8 Conclusion

Dans ce chapitre, nous avons présenté la plateforme SemEx, en particulier la structure de données qu'elle utilise, sa conception, ses interfaces utilisateurs, son architecture modulaire et son implémentation.

Les données sont contenues dans une structure d'index reposant sur trois modèles : un modèle de liens interconnectant un modèle de document qui structure les textes réglementaires et un modèle sémantique, lui-même constitué d'une ontologie décrivant le vocabulaire métier de domaine et d'une base de règles candidates. L'exploitation de cette structure guide la prise en charge des tâches d'acquisition et de maintenance des règles. Cette structure est initialement obtenue par la projection de l'ontologie sur les textes et elle est enrichie au fur et à mesure que de nouvelles règles sont identifiées et normalisées. Son implémentation repose sur des technologies standard du Web sémantique et utilise les fonctionnalités des graphes RDF.

Les interfaces utilisateurs proposées dans SemEx sont définies pour permettre une interactivité avec l'utilisateur, notamment l'expert métier qui doit être guidé dans la mise en œuvre des méthodologies d'identification, de normalisation et de diagnostic des problèmes de maintenance des règles, en particulier dans leur phases manuelles. Les phases automatiques de ces méthodologies quant à elles sont implémentées dans des modules qui mettent à la disposition de l'expert, via les interfaces, des outils permettant la prise en charge automatique d'un certain nombre d'étapes du processus. Nous avons ainsi présenté :

- un module d'accès aux données qui exploite et manipule la structure d'index à travers un système de requêtage SPARQL exécuté dans un moteur sémantique (CORESE) ;
- un module d'identification des règles qui implémente les fouilles syntaxique à base de mots-clés, de patrons linguistiques applicables sur le contenu brut ou sur l'arbre syntaxique des phrases de la base documentaire (modèle de document), et de patrons sémantique utilisant l'ontologie et ses liens avec les textes et la base de règles ;
- un module de normalisation qui implémente pour le moment les opérations de normalisation lexicale, contextuelle et syntaxique (les autres étant prises en charge manuellement par l'expert). Ce module repose essentiellement sur la définition de requêtes SPARQL et de patrons de transformations traduisant des règles non normalisées en des règles normalisées ;
- enfin un module de maintenance permettant de diagnostiquer les problèmes d'incomplétude, d'incohérence et de mise à jour des règles. Il implémente aussi des requêtes SPARQL pour faire cela et celles-ci reposent sur la traçabilité des règles et sur les calculs de contexte de règles et de similarité sémantique entre les règles.

SemEx est implémenté sous forme de plugin Eclipse RCP [12], ce qui favorise la mise sur pied d'interfaces utilisateur conviviales et surtout la modularisation de son architecture, et lui confère une qualité d'application "riche" facilement déployable. Son code source est en Java et l'ensemble des outils qu'elle intègre sont soit dans ce langage, soit s'interfacent avec lui.

12. http ://www.eclipse.org/home/categories/rcp.php

# Troisième partie

# Expérimentations et Perspectives sur SemEx

# Chapitre VII

# Expérimentations

Dans ce chapitre, nous expérimentons les méthodologies que SemEx met en œuvre pour les tâches d'identification des règles à partir des textes réglementaire, et pour leur normalisation. Ces textes sont ceux des corpus AAdvantage et Audi présentés dans l'introduction et décrivant respectivement le programme de fidélisation d'American Airlines et les tests de qualité des ceintures de sécurité des voitures du constructeur Audi. Sur ces deux textes nous appliquons les méthodologies d'identification et de normalisation. Les résultats permettent d'analyser leur apport et leur utilité.

L'analyse consiste à observer le comportement de nos méthodologies sur chacun des corpus afin de voir les différentes stratégies à mettre en place pour améliorer les techniques de fouilles de règles, le processus de normalisation, ainsi que l'interaction avec l'expert.

Pour l'analyse des cas d'usage, nous avons pu travailler et collaborer avec John Hall, membre fondateur du Business Rule Group, participant aux travaux de l'OMG, du W3C, aux comités scientifiques des conférences BR (Business Rules) et AICOL (AI approaches to the Complexity of Legal Systems). Il nous a aidé à identifier les règles dans les textes.

## VII.1 Description des données initiales

### VII.1.1 Corpus AAdvantage

Ce corpus transcrit dans une dizaine de pages est constitué de 243 phrases et décrit les conditions d'attribution et d'utilisation des points de fidélité que gagne un client lorsqu'il achète les services de American Airlines ou d'entreprises alliées. C'est un texte réglementaire dans

| Concepts | Nombres d'occurrences |
|---|---|
| Upgrade | 105 |
| Mile | 70 |
| Award | 68 |
| Mileage credit | 66 |
| AAdvantage member | 59 |
| Travel | 56 |
| Ticket | 51 |
| American Airline | 49 |
| Flight | 44 |
| Account | 27 |

**Tableau VII.1** – Concepts les plus représentatifs du corpus AAdvantage.

lequel sont représentées les différentes catégories de règles métier [1] : les règles structurelles (1), les règles opératoires (2), les règles dérivationnelles (3) (comme nous l'avons vu plus haut) mais aussi des informations pour aider le client à comprendre et utiliser ces points. Ces dernières ne sont pas des règles métier et ne nous intéressent pas ici.

(1) *Accrued mileage credit and award tickets do not constitute property of the member.*
(2) *You may request mileage credit for past, eligible transactions up to 12 months from the transaction date.*
(3) *On connecting flights, you'll receive mileage credit for each segment of your trip.*

Pour ce corpus, nous avons utilisé une ontologie décrivant le vocabulaire métier du domaine. Cette ontologie est constituée d'une hiérarchie de 221 concepts, de 76 relations conceptuelles et 26 individus. Parmi tous ces éléments conceptuels, seuls 152 sont associés à des listes de termes métier synonymes à partir desquels la projection de l'ontologie sur un corpus peut être effectuée. La moyenne du nombre de termes métier par liste est de 1,2 soit 1 à 2 termes par concept (ou relation ou instance). Le corpus est annoté au regard de cette ontologie ; les éléments conceptuels les plus représentés sont décrits dans le tableau VII.1.

1. Particularité de ce document, comme ces catégories sont relatives au point de vue de l'entreprise elles demandent une interprétation particulière. Par exemple, un délai de demande d'un an pour le client devient pour l'entreprise une obligation de recevoir les demandes faites dans les délais.

| Concepts | Nombres d'occurrences |
|---|---|
| Strap | 66 |
| Test | 55 |
| Seat belt | 22 |
| Buckle, Adjusting device | 12 |
| Regulation, Temperature | 11 |
| Low temperature chamber, Device | 8 |
| Atmosphere, Conditionning | 7 |
| Restraint system, Dynamic test | 6 |
| Micro-slip test, Humidity, Dimension | 5 |

**Tableau VII.2** – Concepts les plus représentatifs du corpus Audi.

### VII.1.2 Corpus Audi

Ce corpus forme un document de 130 pages décrivant les tests de qualité à effectuer par le constructeur Audi sur les ceinture de sécurité. Nous nous sommes intéressés principalement à une partie décrivant les conditions de mise en œuvre de ces tests. Il s'agit du Chapitre 7 du document qui est constitué d'une dizaine de page construites autour de 138 phrases. Les règles opératoires (2) sont les plus représentées, suivies des règles structurelles (1), les règles dérivationnelles sont presque inexistantes dans cette partie.

(1) *Any wetting agent suitable for the fibre under test may be used.*
(2) *When the dynamic test has been carried out for a type of vehicle it need not be repeated for other types of vehicle where each anchorage point is less than 50 mm distant from the corresponding anchorage point of the tested belt.*

L'ontologie de domaine que nous utilisons pour ce corpus est plus petite que la précédente et est constituée d'une hiérarchie de 78 concepts, de 15 relations conceptuelles et de 30 individus. 99 de ces éléments conceptuels sont associés à des listes de termes métier synonymes avec une moyenne de 1,14 (1 à 2) termes par élément. Le corpus est aussi annoté au regard de cette ontologie et les éléments conceptuels les plus représentés sont décrits dans le tableau VII.2.

## VII.2 Tests de la méthodologie d'identification

Nous avons testé une forme faible de fouille syntaxique, basée sur les mots-clés (qui sont des mots grammaticaux), et la fouille sémantique afin d'identifier les phrases caractéristiques des

règles. Nous avons utilisé les corpus AAdvantage et Audi et comparé les résultats au diagnostic de l'expert métier.

La fouille syntaxique que nous testons repose sur la recherche de mots-clés, ce qui permet de rechercher dans le contenu brut. Les patrons syntaxiques au sens plein, i.e. qui s'appliquent sur les arbres syntaxiques, sont plus difficiles à intégrer : les analyseurs syntaxiques sont en général entraînes sur des corpus journalistiques, et leur utilisation sur des textes réglementaires n'a commencé que récemment (voir [E. et al., 2012]). Les problèmes de syntaxe spécifiques à ces textes doivent d'abord être résolus. La fouille sémantique quant a elle est effectuée à partir des concepts (relations ou instances) les plus descriptifs des règles candidates déjà identifiées.

Dans l'étude, les règles sont déjà connues et marquées dans le corpus. Il s'agit de savoir si nos indices permettent de discriminer les règles des non-règles. Autrement dit, nous nous posons la question : peut-on construire pour cet échantillon un oracle basé sur ces indices qui reconnaisse toutes les règles sans trop d'excès (il reste alors quelques phrases à éliminer), ou bien qui ne reconnaisse que des règles mais en nombre significatif (il reste à retrouver les quelques manquantes). Bien entendu, pour une situation réelle, où les règles ne sont pas connues à l'avance, on ne dispose pas de cet oracle, sauf si l'on sait caractériser des types de textes. Mais regarder si l'oracle peut fonctionner est déjà une première étape avant de se demander comment le lier au type de texte.

### VII.2.1 Corpus AAdvantage

Les 243 phrases sont manuellement marquées par l'expert qui trouve 95 règles. Nous nous intéressons aux mots-clés avant d'utiliser les concepts du domaine.

**mots-clés** Nous pouvons voir dans le tableau VII.3 les différents mots-clés pour lesquels nous avons des résultats. Il y a des mots-clés à un mot (”if”), à un groupe de mots successifs (”at least one”) et à un groupe de mots éloignés (”neither...nor”). Les mots-clés ont été utilisés en cascade, chacun sélectionnant de nouvelles phrases parmi celles qui ne le sont pas encore. Le total de la seconde colonne donne donc les 195 phrases qui contiennent au moins un mot-clé. Pour chacun, nous avons aussi les mots-clés qui apparaissent en même temps que lui dans les phrases sélectionnées. Le nombre total de phrases où il apparaît figure en dernière colonne.

Nous pouvons remarquer que les mots-clés ”or”, ”and”, ”may”, ”if” sont les plus représentés. cette fouille sélectionne 195 phrases sur les 243 du texte. On a en moyenne 1.43 mot-clé par phrase. Après comparaison avec les phrases identifiées comme règles métier candidates par l'expert, seules 80 phrases sont validées sur 195, couvrant toutes les règles. La sélection par

| mot-clé | # de phrases sélectionnées | mots-clés co-occurrents | # phrases où il apparaît |
|---|---|---|---|
| each | 18 | at least one (2), or (2), if (2), may (1) | 18 |
| some | 7 | or (2), if (1), may (1), shall (1) | 7 |
| can | 8 | more than one (1), or (1), may (1) | 8 |
| and | 61 | or (24), if (4), must (3), may (13) | 61 |
| or | 47 | if (5), neither...nor (1), must (5), may (12), shall (1) | 76 |
| if | 13 | may (2) | 25 |
| then | 2 | | 2 |
| must | 5 | | 13 |
| may | 16 | | 46 |
| It is necessary | 1 | | 1 |
| neither...nor | 1 | | 2 |
| for | 16 | | 16 |
| at least one | | | 2 |
| shall | | | 1 |
| more than one | | | 1 |
| Total | 195 | | 279 |

**Tableau VII.3** – AA – Fouille des mots-clés sur l'ensemble des phrases.

la présence d'au moins un mot-clé n'est pas un critère très significatif dans AAdvantage : la proportion de règles dans les phrases sélectionnées est 41%, contre 31% dans celles qui ne le sont pas.

Nous avons regardé si certains mots-clés ou certaines configurations de mots-clés étaient particulièrement significatifs. Pour vérifier cela, nous mesurons la présence des mots-clés dans les 80 règles en comportant (tableau VII.4). Nous ajoutons, là où l'échantillon le permet, leur pertinence sélective (nombre de règles où ils figurent / nombre de phrases où ils figurent). Ce nombre se compare à une pertinence de base de 80 règles sur 195 phrases (41%) ou 95 règles sur 243 phrases (39%).

On constate qu'aucun mot-clé ne donne à lui seul d'indication décisive – 'and', qui est le plus fréquent, est exactement à la moyenne et donc ne donne aucune information, ce qui n'est pas surprenant étant donné la multiplicité de ses usages. Les plus significatifs sont 'for' et 'must' qui à eux deux sélectionnent 29 phrases dont 20 règles. Même en ajoutant les phrases sélectionnées par 'each', on ne trouve pas la moitié des règles. Il y a donc lieu de regarder si la fouille sémantique peut améliorer les résultats.

| mot-clé | # de règles sélectionnées | mots-clés co-occurrents | # règles où il apparaît |
|---|---|---|---|
| each | 10 | at least one (2), or (1), may (1), if (1), for (2) | 10 (55%) |
| some | 0 | | 0 |
| can | 4 | more than one (1), or (1) | 4 (50%) |
| and | 23 | or (15), if (3), must (3), may (8) | 23 (38%) |
| or | 18 | if (3), neither...nor (1), must (4), may (7) | 35 (46%) |
| if | 6 | may (1) | 13 (52%) |
| then | 1 | | 1 |
| must | 2 | | 9 (69%) |
| may | 6 | | 23 (46%) |
| It is necessary | 1 | | 1 |
| neither...nor | 0 | | 1 |
| for | 9 | | 11 (69%) |
| shall | 0 | | 0 |
| more than one | 0 | | 1 |
| Total | 80 | | 132 |

**Tableau VII.4** – AA – Fouille des mots-clés sur les règles candidates.

**Fouille sémantique** Dans la fouille sémantique, il s'agit d'identifier les phrases pertinentes à partir des concepts du domaine. Nous avons dans un premier temps croisé les mots-clés avec les principaux concepts représentatifs du corpus AAdvantage (cf. tableau VII.1). On trouve dans le tableau VII.5 le nombre de phrases du corpus où co-occurrent un mot-clé donné et un concept donné, et (entre parenthèses) le nombre de règles satisfaisant la même condition.

| Concepts/mot-clé | each | can | and | or | if | must | may | for | Total |
|---|---|---|---|---|---|---|---|---|---|
| Upgrade | 6 (4) | 0 (0) | 14 (5) | 12 (6) | 5 (2) | 0 (0) | 3 (0) | 3 (0) | 43/13 |
| Mile | 3 (2) | **1 (1)** | 20 (10) | 8 (3) | **3 (3)** | 0 (0) | 2 (1) | 1 (0) | 38/20 |
| Award | 2 (1) | 3 (2) | 19 (9) | 14 (6) | **2 (2)** | 2 (1) | 5 (2) | 4 (3) | 51/25 |
| Mileage credit | 7 (3) | **3 (3)** | 16 (7) | 14 (7) | **1 (1)** | 1 (0) | **6 (6)** | 7 (4) | 55/31 |
| AAdvantage member | 5 (2) | 0 (0) | 17 (7) | 13 (6) | **1 (1)** | 2 (1) | 5 (3) | 4 (2) | 48/22 |
| Travel | **1 (1)** | **1 (1)** | 13 (6) | 6 (3) | **3 (3)** | 1 (0) | 4 (2) | **2 (2)** | 31/18 |
| Ticket | **1(1)** | 3 (2) | 15 (9) | 12 (7) | 3 (1) | **1 (1)** | 3 (2) | 6 (4) | 44/27 |
| American airline | **1 (1)** | **2 (2)** | 23 (15) | 12 (2) | 1 (0) | 2 (1) | 2 (1) | **2 (2)** | 45/24 |
| Flight | 5 (4) | **2 (2)** | 10 (4) | 7 (3) | 3 (2) | **1 (1)** | 5 (3) | **3 (3)** | 36/22 |
| Account | **2 (2)** | 0 (0) | 4 (3) | 3 (2) | **2 (2)** | 1 (0) | 1 (0) | 0 (0) | 13/9 |
| Total | 33/21 | 15/13 | 151/75 | 101/45 | 24/17 | 11/5 | 36/20 | 32/20 | |

**Tableau VII.5** – AA – Fouille des mots-clés suivant leurs co-occurrences avec les termes métier les plus fréquents du texte sur l'ensemble des phrases et sur les règles candidates.

On voit dans la dernière colonne que la pertinence sélective globale d'un concept (indépendamment du mot-clé associé) atteint au mieux 61% (pour 'flight' et 'ticket') et on peut calculer qu'elle est de 52% en moyenne.

On a marqué en gras les combinaisons qui ne sélectionnent que des règles. La plus significative est 'mileage credit' + 'may', puis 'mileage credit' + 'can', 'mile' + 'if', 'travel' + 'if', 'flight' + 'for'. Les effectifs sont toutefois trop faibles pour en tirer des conclusions solides et nous avons seulement une piste à poursuivre.

En ce qui concerne les phrases qui n'ont pas de mot-clé, nous avons regardé le pouvoir de sélection des concepts du domaine. Les résultats figurent dans le tableau VII.6. Les concepts du domaine les plus représentés ont permis de sélectionner 24 phrases dont 15 sont des règles métier.

| Concept | # de phrases sélectionnées |
|---|---|
| mileage credit | 2 |
| AAdvantage member | 2 |
| flight | 1 |
| ticket | 3 |
| American Airline | 4 |
| Travel | 1 |
| AAdvantage participant | 1 |
| Upgrades | 9 |
| qualifying activity | 1 |

**(a)** AA – Fouille sémantique sur l'ensemble des phrases

| Concept | # de règles où il figure |
|---|---|
| mileage credit | 1 |
| AAdvantage member | 2 |
| flight | 1 |
| ticket | 3 |
| American Airline | 4 |
| Travel | 1 |
| Upgrades | 2 |
| qualifying activity | 1 |

**(b)** AA – Fouille sémantique sur les règles métier

**Tableau VII.6** – AA – Fouille sémantique sur les phrases sans mot-clé.

### VII.2.2 Corpus Audi

Nous avons dit que ce corpus est linguistiquement assez différent du précédent, en ce qu'il s'adresse à des spécialistes. On va constater très vite que les résultats de la fouille permettent des méthodes d'identification plus simples.

**Mots-clés** La fouille par mots-clés sélectionne 125 phrases sur les 138 de départ, dont les 100 règles. Le tableau VII.7 présente le détail des résultats. Ici encore, on a en seconde colonne les résultats avec élimination successive des phrases déjà sélectionnées (l'ordre est arbitraire) et dans la dernière colonne les chiffres sans élimination. Le simple critère de présence d'un mot-clé

| mot-clé | # de phrases sélectionnées | mots-clés co-occurrents | # phrases où il apparaît |
|---|---|---|---|
| each | 13 | for (4), need not (1), and (4), or (1), shall (11) | 13 |
| for | 41 | and (16), or (8), if (3), then (2), may (5), shall (23) | 45 |
| can | 1 | shall (1) | 1 |
| at least | 3 | and (3), then (1), shall (3) | 3 |
| and | 23 | or (3), if (1), then (2), shall (16) | 46 |
| or | 7 | if (2), must (1), shall (5) | 19 |
| if | 2 | shall (2) | 8 |
| shall | 35 | | 96 |
| need not | | | 1 |
| then | | | 5 |
| may | | | 5 |
| must | | | 1 |
| Total | 125 | | 237 |

**Tableau VII.7** – Audi – Fouille des mots-clés sur l'ensemble des phrases.

reste peu sélectif : il élimine à bon escient, mais seulement 10% des phrases, et 20% de celles qui restent ne sont pas des règles. Ceci dit, il faut aussi constater que les mots-clés n'ont pas tous la même importance quantitative : 40% des occurrences sont dues à 'shall', et si l'on ajoute 'for' et 'and', on atteint 79%.

Nous avons ensuite regardé si, dans ce corpus, nous pouvions mieux caractériser les règles. Nous avons pour cela comparé le tableau précédent à la même fouille faite uniquement sur les règles (tableau VII.8). Les pourcentages de la dernière colonne comparent le nombre d'apparitions dans les règles et dans le corpus entier.

Un point s'impose à la lecture de ce tableau : le marqueur 'shall' apparaît dans 91 règles sur les 100 qu'en compte le texte, alors que ce même marqueur a en tout 96 occurrences. Si l'on ajoute que 'if' est un marqueur sûr qui donne 6 règles supplémentaires, le choix de ces deux marqueurs sélectionne 102 phrases contenant 97 des 100 règles. L'erreur portant sur des nombres trop faibles pour qu'une amélioration nous renseigne sur l'apport du vocabulaire de domaine, nous n'avons pas poussé plus loin.

### VII.2.3 Bilan

Il reste de l'expérience que la fouille syntaxique peut avoir de très bonnes performances pour l'identification avec des moyens simples - c'est le cas du corpus Audi, dont pourtant le texte est

difficile. La question du choix des filtres syntaxiques selon le texte reste ouverte, mais on peut raisonnablement espérer que cet outil se révèle efficace pour explorer de nouveaux textes. Dans aucune de nos expériences la fouille sémantique n'est apparue très sélective. Ceci n'exclut pas son emploi, mais indique qu'elle serait surtout utile en complément : soit pour raffiner sur la fouille syntaxique, soit en interaction avec l'expert.

## VII.3 Tests de la méthodologie de normalisation

Il s'agit, à partir des premières versions des règles, supposées maintenant identifiées dans l'étape précédente, de les soumettre au processus de normalisation chargé de réduire les complexités de la langue naturelle. Nous analysons ici le comportement des opérations de normalisation lexicale, contextuelle, et syntaxique sur les corpus AAdvantage et Audi. Nous n'avons pas pris en compte dans nos expériences la normalisation sémantique qui enrichit le domaine en créant de nouveaux termes métier.

Les tableaux VII.10, VII.9, VII.11, VII.12, VII.13 montrent les résultats des opérations de normalisation effectuées sur les deux corpus. Le premier récapitule le nombre de règles candidates initiales (obtenues à l'étape d'identification) et son évolution après normalisation. Le deuxième tableau décrit les proportions d'utilisation de chaque type de normalisation (lexicale, contextuelle ou syntaxique) par rapport d'une part ($\alpha$) à l'ensemble des règles et d'autre part

| mot-clé | # de règles sélectionnées | mots-clés co-occurrents | # règles où il apparaît |
|---|---|---|---|
| each | 12 | for (2), need not (1), and (4), or (1), shall (10) | 12 (92%) |
| for | 28 | and (18), or (6), if (3), then (2), may (2), shall (25) | 30 (66%) |
| can | 1 | shall (1) | 1 |
| and | 16 | or (3), if (1), then (2), shall (13) | 38 (82%) |
| or | 6 | if (2), must (1), shall (5) | 16 (84%) |
| if | 2 | shall (2) | 8 (100%) |
| shall | 35 | | 91 (95%) |
| need not | | | 1 |
| then | | | 4 |
| may | | | 2 |
| must | | | 1 |
| Total | 100 | | 204 |

**Tableau VII.8** – Audi – Fouille des mots-clés sur les règles candidates.

($\beta$) au total des opérations, tout type confondu, appliquées sur toutes ces règles. Et enfin les trois derniers qui décrivent quantitativement l'impact relatif de chacun des types d'opération sur les mots et les phrases

Nous analysons ci-dessous, au regard de ces tableaux, le fonctionnement du processus de normalisation sur les règles candidates des corpus AAdvantage et Audi. Dans nos tests, nous appliquons successivement les normalisations lexicale, contextuelle et syntaxique.

| Corpus | règles candidates initiales | règles candidates finales |
|---|---|---|
| AAdvantage | 95 | 134 |
| Audi | 100 | 120 |

**Tableau VII.9** – Nombre de règles candidates obtenues après normalisation.

| Type de Normalisation | AAdvantage $\alpha$ | AAdvantage $\beta$ | Audi $\alpha$ | Audi $\beta$ |
|---|---|---|---|---|
| Lexicale | 65,3% (62/95) | 28,4% | 61% (61/100) | 27,9% |
| Contextuelle | 64,2% (61/95) | 27,9% | 57% (57/100) | 26,2% |
| Syntaxique (structuration) | 68,4% (65/95) | 29,9% | 58% (58/100) | 26,6% |
| Syntaxique (décomposition) | 31,6% (30/95) | 13,8% | 42% (42/100) | 19,3% |

**Tableau VII.10** – $\alpha$ = Nbr de transformations d'un type donné / Nbr total de CR initiales ; $\beta$ = Nbr de transformations d'un type donné / Nbr total de transformations.

### VII.3.1 Corpus AAdvantage

En ce qui concerne le corpus AAdvantage, on constate que :

- Au niveau lexical, la normalisation concerne 65,3% des règles candidates soit 62 règles sur les 95 initialement identifiées dans AAdvantage. Elle a permis de remplacer dans toutes les règles candidates chacune des occurrences d'un terme métier qui n'est pas le terme préféré associé au concept (relation ou instance) annotateur, par une occurrence de ce dernier terme. Le tableau VII.11 montre que 80 occurrences de termes métier sont dans ce cas, soit une moyenne de 1,26 occurrences par règle candidate concernée. Cette moyenne s'explique par le fait que plus de 77% de ces règles (47) ne contiennent qu'une seule occurrence de termes à remplacer, seules quelques unes (14) en contiennent 2 ou 3.
- Au niveau contextuel, la normalisation concerne 64,2% des règles. Elle a porté sur 70 mots. Ce sont essentiellement des mots grammaticaux (”you” et ”your”, ”this”, ”that”, ”other”) renvoyant à des termes métier très précis comme AAdvantage member pour le

| Corpus | # règles concernées | # Occurrences termes remplacées | # de règles dont 1, 2, 3 termes sont remplacés |
|---|---|---|---|
| AAdvantage | 62 | 80 | 48, 10, 4 |
| Audi | 61 | 77 | 47, 12, 2 |

**Tableau VII.11** – Normalisation lexicale.

| Corpus | # règles concernées | # occurrences mots à contextualiser |
|---|---|---|
| AAdvantage | 61 | **70** (you (29), your (28), this (3), that (2), mileage credit (5), other (3)) |
| Audi | 57 | **109** (it (7), this (15), 14 (that), sample (16), the test (29), Annex (11), paragraph (17)) |

**Tableau VII.12** – Normalisation contextuelle.

| Corpus | Structuration<br># règles concernées (type) | Décomposition<br># règles concernées (type) |
|---|---|---|
| AAdvantage | 65 (B if A ; B, A ; B when A ; B for each A) | 29 (or ; and ; neither nor ; which ; who) |
| Audi | 30 (B if A ; B, A) | 42 (or ; and ; which ; beetween) |

**Tableau VII.13** – Normalisation syntaxique.

”you”, et rarement des relations cachées, en fait seulement celles qui dépendent du terme ”mileage credit”. On a une moyenne de 1,2 mots par règle concernée dont il faut rétablir le contexte.

- Au niveau syntaxique, la normalisation concerne la totalité des règles. Elle permet de restructurer syntaxiquement une règle candidate ou de la décomposer en plusieurs autres règles candidates. La structuration est plus représentée dans ce corpus dans la mesure où elle concerne 68,4% des règles ; la décomposition n'a été effectuée que sur 30,6% des règles mais le nombre de règles augmente toutefois de 39 règles. Les cas de structuration sont le plus souvent décrits dans ce corpus par des formules du genre B if A ; B, A ; B when A ; B for each A qui peuvent devenir If A then B ; When A then B ; For each A then B. La décomposition quant à elle est justifiée par la présence d'opérateurs logiques “and” et “or” ou alors de conjonctions comme ”which”, ”who”, ”neither nor”.

Nous pouvons aussi remarquer que, en nombre d'opérations, la normalisation syntaxique a une plus grande part (38,5%) que les normalisations lexicale (32,8%) et contextuelle (28,7%) dans ce corpus mais tout de même chaque type de normalisation concerne au moins 60% des règles et donc nous pouvons dire que ce processus est utile pour réduire les complexités du langage naturel.

### VII.3.2 Corpus Audi

Pour le corpus Audi nous avons une utilisation des opérations de normalisation à peu près similaire à AAdvantage. La normalisation syntaxique y est aussi plus représentée (45% des transformations) que les autres et elles couvrent chacune plus de 55% des règles. On peut ainsi constater que :

- au niveau lexical, la normalisation porte sur 61% des règles initiales et 77 occurrences de termes ont fait l'objet de remplacement par des termes préférés, soit une moyenne de 1,3 occurrences par règle candidate ;
- au niveau contextuel, la normalisation concerne 57% des règles et porte sur plus de mots contextuels que dans AAdvantage, soit une moyenne importante : 1,9 termes par règle à remplacer. Parmi ces mots nous avons des mots grammaticaux (36), les termes génériques à spécifier (”test” (29) et ”sample” (16)) mais aussi des clés de références du genre ”Annex” (11) et ”paragraph”(17) ;
- au niveau syntaxique, la normalisation concerne aussi la totalité des règles. Sur ce corpus la décomposition est la plus représentée (42%) par rapport à la structuration qui s'applique sur 30% des règles. Cette dernière permet de structurer essentiellement des règles sous forme (B if A ou B, A) alors que la décomposition ne porte que sur les mots ”and”, ”or”, ”which”, et ”between”.

## VII.4 Conclusion

Dans ce chapitre nous avons pu observer le comportement de nos méthodologies d'identification et de normalisation des règles contenues dans les corpus AAdvantage et Audi.

Le résultat de l'identification dépend du style des textes que nous manipulons. Les techniques de fouille lexicale et sémantique nécessitent de vraies stratégies d'analyse des textes pour espérer avoir le plus de règles dans les phrases sélectionnées et proposées à l'expert. La fouille par mots-clés montre que s'ils sont utilisés de façon aveugle, ils fournissent un résultat bruité. Dans AAdvantage par exemple aucun mot-clé n'est vraiment indicateur, seuls quelques uns (”for”, ”must”, ”can”, ”each”) arrivent à fournir de bons résultats mais ils n'apparaissent que très rarement dans ce corpus et ne fournissent même pas la moitié des règles. Nous avons pensé à utiliser les co-occurrences (mot-clé, terme métier) les plus descriptives de règles et le résultat n'est pas plus significatif. En effet, si dans AAdvantage ces configurations permettent de capturer 61 règles sur les 80, rien ne nous permet de prédire quelles sont les bonnes configurations pour un texte donné. A l'inverse, sur un corpus plus spécialisé comme Audi, la fouille

est plus efficace. Sur la base d'un nombre réduit de mots-clés et de combinaisons de mots-clés, nous arrivons à retrouver la plupart des règles. La conclusion essentielle est que la fouille peut être un réel apport dans certains types de textes, mais qu'il faut des connaissances spécifiques au type de texte.

Les résultats de la normalisation correspondent bien à ce que nous attendions de celle-ci. Quel que soit le texte (AAdvantage ou Audi), un nombre important de règles (plus de 50%) justifie des opérations de normalisation. On peut en tirer quelques idées d'opérations utiles et les travailler pour mettre en œuvre une interactivité efficace avec l'expert, soit pour le guider en identifiant les tournures qu'il faut normaliser, soit même pour lui montrer comment le faire. Pour l'instant, notre expérience permet tout simplement de montrer la nécessité de normalisation, c'est à l'expert d'être "expert" pour faire ce travail.

Nous n'avons pas pu expérimenter la méthodologie de maintenance dans ce chapitre, faute de temps et, encore plus, de données. Nous en restons donc sur ce point à la définition de requêtes SPARQL pertinentes pour certains types de problèmes de maintenance présents dans la littérature.

# Chapitre VIII

# Conclusion générale

Ce mémoire a présenté mes travaux de thèse au terme desquels nous avons réalisé et produit la plateforme SemEx qui vient guider un utilisateur expert-métier dans ses tâches d'acquisition et de maintenance des règles métier notamment à partir de textes réglementaires.

Ma lecture de l'état de l'art montre que ces tâches sont décrites comme parties intégrantes du développement des applications informatiques dans lesquelles les règles métier sont appliquées notamment dans les architectures de type MDA où elles sont définies, implémentées, et maintenues. Cependant, cette configuration ne prend pas en compte dans l'application les textes réglementaires eux-mêmes alors que les règles sont très souvent formulées initialement en langage naturel (LN) par des experts métier qui ne visent généralement aucune informatisation mais uniquement la description des besoins de l'entreprise. Les quelques travaux qui ont proposé des techniques pour identifier automatiquement les règles dans les textes et les formaliser butent sur les complexités du langage naturel. Ce qui suppose alors une réelle collaboration avec un expert métier sur toutes les étapes où son intervention est obligatoire pour résoudre les problèmes.

J'ai organisé alors la gestion des deux tâches dans SemEx sous forme de méthodologies souples et interactives qui permettent de mieux structurer les opérations à effectuer dans chaque tâche ainsi que leurs enchaînements. Ces méthodologies sont implémentées chacune dans un module qui interagit avec une interface utilisateur interactive dans SemEx. Dans chaque module sont intégrés des outils développés par ailleurs (Tree-tagger, CORESE, Stanford Parser, Tregex, Tsurgeon, etc) et permettant de mettre en œuvre le fonctionnement des méthodologies qui sont :

- L'identification des règles à partir des textes, basée sur des techniques de fouille textuelle syntaxique et sémantique via des patrons linguistiques construits suivant un ensemble de mots-clés caractéristiques de règles et de termes métier de domaine.
- La normalisation des règles qui permet de réduire les complexités du langage naturel s'applique à la fois au lexique d'une règle et à sa structure syntaxique ; elle tient compte

aussi de son contexte, de sa sémantique. La normalisation s'arrête aux portes de la formalisation, elle permet de mettre les règles dans une forme facile à formaliser.
– La maintenance des règles qui permet de diagnostiquer les problèmes d'incomplétude, d'incohérence et d'évolution qui peuvent affecter les règles après extraction et nécessiter leur mise à jour.

Cette thèse a permis de réfléchir à la mise en place de la plate-forme, de le faire en ayant conscience du cadre applicatif, de modéliser les données qu'elle manipule de façon à simplifier leur accès par les traitements, et de l'implémenter. Et sur ce dernier point, il a fallu faire des choix techniques. Je me suis appuyé d'une part sur Java et Eclipse pour développer une application Client Riche (RCP) reposant sur une modularisation des tâches et qui est facile à déployer, d'autre part sur des technologies du Web sémantique (RDF, RDFs, RDFa, OWL) qui me permettent de représenter et structurer mes données dans un graphe conceptuel RDF et d'utiliser ses fonctionnalités de navigation via le langage de requête SPARQL et le moteur sémantique d'inférence CORESE.

J'ai donc développé SemEx comme un logiciel libre et l'ai mis en ligne avec un manuel d'utilisateur (voir Annexe). La plate-forme a fait l'objet de plusieurs démonstrations notamment dans les réunions plénières du projet OntoRule mais aussi dans des conférences scientifiques. L'aspect utilisateur a été testé par l'équipe Ergonomie de IBM qui m'a fait des retours utiles. J'ai aussi entrepris d'observer le comportement des méthodologies suggérées sur de réels cas d'usages comme AAdvantage et Audi. Cela m'a permis de voir que la plateforme peut avoir une réelle utilité et que l'identification fonctionne bien avec des textes ayant un style spécialisé, de même que la normalisation des règles. Cela ne suffit pas pour tirer des conclusions définitives avec uniquement ces deux corpus et le travail qui reste à faire est encore important.

J'ai le sentiment d'avoir produit quelque chose pour la recherche ne serait-ce que pour avoir pointé du doigt un vrai problème de recherche, aidé à identifier comment prendre en charge ce problème et proposé un début de solution face à celui-ci. J'ai beaucoup appris durant cette thèse, elle m'aura appris à maîtriser un sujet qui me passionnait et qui me passionne encore plus aujourd'hui. Mon envie, de continuer sur ces questions d'acquisition et de représentation des connaissances et surtout sur ce nouveau monde très en vogue depuis quelques années qu'est le Web sémantique, en sort grandie.

Malgré le travail accompli, j'aurais aimé avoir plus de temps pour aller jusqu'au bout du développement de SemEx, jusqu'au bout du travail méthodologique afin de prendre en compte de plus en plus les besoins de l'expert face aux tâches ; j'aurais aimé faire un peu plus d'expérimentation pour améliorer mes choix techniques de méthodes, de conception et de modélisation, et participer à des campagnes d'évaluation afin de mieux faire connaître SemEx. Mais tout

cela n'est que partie remise, j'aurais certainement l'occasion de poursuivre dans mes travaux postdoctoraux. Sinon j'espère que d'autres chercheurs pourront continuer ce travail.

# Références bibliographiques


[Abacha and Zweigenbaum, 2010] Abacha, A. B. and Zweigenbaum, P. (2010). Annotation et interrogation sémantiques de textes médicaux. In *Atelier Web Sémantique Médical à IC 2010*, pages 61–70, Nîmes. 47

[Adida and Birbeck, 2008] Adida, B. and Birbeck, M. (2008). RDFa primer : Bridging the human and data webs. W3c working group note, W3C. 49

[Adida et al., 2008] Adida, B., Birbeck, M., McCarron, S., and Pemberton, S. (2008). Rdfa in xhtml : Syntax and processing. *A collection of attributes and processing rules for extending XHTML to support RDF*. 49

[Aidas and Olegas, 2009] Aidas, S. and Olegas, V. (2009). Business rules based agile ERP systems development. *Informatica*, 20 :439–460. 15, 28

[Amardeilh, 2007] Amardeilh, F. (2007). *Web Sémantique et Informatique Linguistique : propositions méthodologiques et réalisation d'une plateforme logicielle*. Informatique, Paris 10 - MODYCO. 46, 48

[Amardeilh et al., 2005] Amardeilh, F., Laublet, P., and Minel, J.-L. (2005). Document annotation and ontology population from linguistic extractions. In *Proceedings of the 3rd international conference on Knowledge capture (K-CAP '05)*, pages 161–168, New York, NY, USA. ACM. 46, 47

[Angles et al., 2004] Angles, R., Gutierrez, C., and Hayes, J. (2004). RDF query languages need support for graph properties. Technical report, Universidad de Chile. 53, 54

[Antoniou et al., 2005] Antoniou, G., Franconi, E., and Harmelen, F. V. (2005). Introduction to semantic web ontology languages. In *Reasoning Web, Proceedings of the Summer School, Malta, 2005. Number 3564 in Lecture Notes in Computer Science*. Springer. 50

[Assem et al., 2006] Assem, M., Malaisé, V., Miles, A., and Schreiber, G. (2006). A method to convert thesauri to SKOS. In Sure, Y. and Domingue, J., editors, *The Semantic Web : Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 95–109. Springer Berlin Heidelberg. 52

[Aussenac-Gilles et al., 2008] Aussenac-Gilles, N., Després, S., and Szulman, S. (2008). The terminae method and platform for ontology engineering from texts. In Buitelaar, P. and

Cimiano, P., editors, *Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text*, pages 199–223. IOS Press. 47, 52

[Baget et al., 2005] Baget, J.-F., Canaud, E., Euzenat, J., and Hacid, M.-S. (2005). Les langages du web sémantique. *Information - Interaction and Intelligence (I3) Une Revue en Sciences du Traitement de l'Informa.* Numéro hors série. 48

[Bajec and Krisper, 2005] Bajec, M. and Krisper, M. (2005). Issues and challenges in business rule-based information systems development. In *ECIS*. 15, 25, 28

[Bajwa and Lee, 2011] Bajwa, I. S. and Lee, M. G. (2011). Transformation rules for translating business rules to OCL constraints. In *Proceedings of the 7th European conference on Modelling foundations and applications*, ECMFA'11, pages 132–143, Berlin, Heidelberg. Springer-Verlag. 41

[Bajwa et al., 2011] Bajwa, I. S., Lee, M. G., and Bordbar, B. (2011). SBVR business rules generation from natural language specification. In *AAAI Spring Symposium 2011 Artificial Intelligence 4 Business Agility*, pages 541–545, San Francisco, USA. AAAI. 39, 40

[Barcellini et al., 2012] Barcellini, F., Albert, C., Grosse, C., and Saint-Dizier, P. (2012). Risk analysis and prevention : LELIE, a tool dedicated to procedure and requirement authoring. In Calzolari, N., Choukri, K., Declerck, T., Dogan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. ELRA. 32

[Bechhofer et al., 2002] Bechhofer, S., Carr, L., Goble, C. A., Kampa, S., and Miles-Board, T. (2002). The semantics of semantic annotation. In *On the Move to Meaningful Internet Systems*, DOA/CoopIS/ODBASE Confederated International Conferences, pages 1152–1167, London, UK, UK. Springer-Verlag. 52

[Berzins et al., 2008] Berzins, V., Martell, C., Luqi, and Adams, P. (2008). Innovations in natural language document processing for requirements engineering. In Paech, B. and Martell, C., editors, *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs*, volume 5320 of *Lecture Notes in Computer Science*, pages 125–146. Springer Berlin Heidelberg. 32

[Bizer et al., 2007] Bizer, C., Cyganiak, R., and Heath, T. (2007). How to publish linked data on the web. http ://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/. Revised 2008. Accessed 22/02/2010. 53, 54

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3) :1–22. 54

[Braga et al., 2000] Braga, R. M. M., Werner, C. M. L., and Mattoso, M. (2000). Using ontologies for domain information retrieval. In *Proceedings of the 11 th International Workshop on Database and Expert Systems Applications. IEEE Computer Society*, pages 836–840. IEE Computer Society Press. 47

[Braye et al., 2006] Braye, L., Ramel, S., Grégoire, B., Leidner, S., and Schmitt, M. (2006). State of the art business rules languages. Technical report, Centre de Recherche Public Henri Tudor. 28

[BRG, 2000] BRG (July, 2000). Defining business rules  what are they really ? The Business Rules Group : formerly, known as the GUIDE Business Rules Project - Final Report revision 1.3. 16, 17, 19, 25, 29

[Buitelaar et al., 2011] Buitelaar, P., Cimiano, P., McCrae, J., Montiel-Ponsoda, E., and Declerck, T. (2011). Ontology lexicalization : The lemon perspective. In *9th International Conference on Terminology and Artificial Intelligence*. 52

[Cabot et al., 2010] Cabot, J., Pau, R., and Raventos, R. (2010). From UML/OCL to SBVR specifications : a challenging transformation. *Information Systems*, 35(4) :417–440. 46

[Candido et al., 2009] Candido, Jr., A., Maziero, E., Gasperin, C., Pardo, T. A. S., Specia, L., and Aluisio, S. M. (2009). Supporting the adaptation of texts for poor literacy readers : a text simplification editor for brazilian portuguese. In *Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 34–42, Stroudsburg, PA, USA. Association for Computational Linguistics. 43, 44

[Carroll et al., 1998] Carroll, J., Minnen, G., Canning, Y., Devlin, S., and Tait, J. (1998). Practical simplification of english newspaper text to assist aphasic readers. In *In Proc. of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10. 42, 44

[Ceravolo et al., 2007] Ceravolo, P., Fugazza, C., and Leida, M. (2007). Modeling semantics of business rules. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pages 171 – 176, Cairns. 41

[Chabbat, 1997] Chabbat, B. (1997). *Modélisation Multiparadigme de Textes Réglementaires*. PhD thesis, INSA Lyon. 20

[Chandrasekar et al., 1996] Chandrasekar, R., Doran, C., and Srinivas, B. (1996). Motivations and methods for text simplification. In *In Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING '96*, pages 1041–1044. 42, 45, 139

[Chandrasekar and Srinivas, 1997] Chandrasekar, R. and Srinivas, B. (1997). Automatic induction of rules for text simplification. *Knowl.-Based Syst.*, 10(3) :183–190. 42, 45, 139

[Chniti et al., 2012] Chniti, A., Albert, P., and Charlet, J. (2012). Gestion de la cohérence des règles métier éditées à partir d'ontologies OWL. In *Actes de IC2011*, pages 589–606, Chambéry, France. 22, 30

[Ciravegna and Wilks, 2003] Ciravegna, F. and Wilks, Y. (2003). Designing adaptive information extraction for the semantic web in Amilcare. In S., H. and S., S., editors, *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*, pages 112–127. IOS Press, Springer-Verlag. 49

[Cisternino et al., 2009] Cisternino, V., Corallo, A., Elia, G., and Fugazza, C. (2009). Business rules for semantics-aware business modelling : overview and open issues. *Int. J. Web Eng. Technol.*, 5 :104–131. 28

[Corby et al., 2004] Corby, O., Dieng-kuntz, R., and Faron-zucker, C. (2004). Querying the semantic web with the corese search engine. In *Ecai*, pages 705–709. IOS Press. 54

[Cunningham, 2002] Cunningham, H. (2002). Gate - a general architecture for text engineering. In *Computers and the Humanities, Volume 36*, pages 223–254. 49

[De Belder et al., 2010] De Belder, J., Deschacht, K., and Moens, M.-F. (2010). Lexical simplification. In *Proceedings of Itec2010 : 1st International Conference on Interdisciplinary Research on Technology, Education and Communication,*. 42, 44

[Dill et al., 2003] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., S.Rajagopalan, Tomkins, A., J.A.Tomlin, and Zien, J. (2003). Semtag and seeker : Bootstrapping the semantic web via automated semantic annotation. In *WWW'03*, pages 178–186, Budapest,Hongrie. ACM Press. 49

[Diouf et al., 2007] Diouf, M., Maabout, S., and Musumbu, K. (2007). Génération automatique de règles métier par enrichissement sémantique de modèles. In *INFORSID*, pages 453–468. 15

[Dubauskaite and Vasilecas, 2009] Dubauskaite, R. and Vasilecas, O. (2009). An open issue in business rules based information system development. In *Innovative Infotechnologies for Science, Business and Education*, volume 1. 25, 29, 60

[E. et al., 2012] E., F., S., M., W., P., and A., W., editors (2012). *Semantic Processing of Legal Texts (SPLeT)*. Workshop of ELREC. 154

[Eder et al., 1986] Eder, J., Kappel, G., Tjoa, A. M., and Wagner, R. (1986). Bier - the behaviour integrated entity realtionship approach. In Spaccapietra, S., editor, *Entity-Relationship Approach : Ten Years of Experience in Information Modeling, Proceedings of the Fifth International Conference on Entity-Relationship Approach, Dijon, France, November 17-19, 1986*, pages 147–166. North-Holland. 28

[Fink et al., 2011] Fink, M., El Ghali, A., Chniti, A., Korf, R., Schwichtenberg, A., Lévy, F., Puhrer, J., and Eiter, T. (2011). D2.6 consistency maintenance. In Deliverable, O., editor, *WP2 Business Rules and Ontologies Ownership and Management*. 30

[Gandon, 2008] Gandon, Fabien, L. (2008). *Graphes RDF et leur Manipulation pour la Gestion de Connaissances*. Hdr, Université de Nice Sophia-Antipolis. 53, 54

[Gasperin et al., 2009] Gasperin, C., Specia, L., Pereira, T. F., and Aluisio, S. M. (2009). Learning when to simplify sentences for natural text simplification. In *ENIA 2009 (VII Encontro Nacional de Inteligência Artificial)*. 42

[Guissé et al., 2009] Guissé, A., Lévy, F., Nazarenko, A., and Szulman, S. (2009). Annotation sémantique pour l'indexation de règles métiers. *TIA'09*, 11. 46, 47, 68

[Haase et al., 2004] Haase, P., Broekstra, J., Eberhart, A., and Volz, R. (2004). A comparison of RDF query languages. In *3rd International Semantic Web Conference, ISWC 2004*. 54

[Halle et al., 2006] Halle, B., Goldberg, L., and Zackman, J. (2006). *Business Rule Revolution : Running Business the Right Way*. Happy About. 15, 23, 25, 29

[Halpin, 2004] Halpin, T. (2004). Business rule verbalization. In Doroshenko, A. E., Halpin, T. A., Liddle, S. W., and Mayr, H. C., editors, *ISTA*, volume 48 of *LNI*, pages 39–52. Gesellschaft für Informatik. 30

[Hayes and Gutierrez, 2004] Hayes, J. and Gutierrez, C. (2004). Bipartite graphs as intermediate model for rdf. In *In Proc. of the 3th Int. Semantic Web Conference (ISWC), number 3298 in LNCS*, pages 47–61. Springer-Verlag. 54

[Hayes, 2004] Hayes, P. (2004). RDF Semantics. Recommendation, World Wide Web Consortium. 48, 53

[Huyck and Abbas, 2000] Huyck, C. R. and Abbas, F. (2000). Natural language processing and requirements engineering : a linguistics perspective. In *Proceedings of 1st Asia-Pacific Conference on Software Quality*. 32

[Jackson, 1997] Jackson, M. (1997). The meaning of requirements. *Annals of Software Engineering*, 3 :5–21. 32

[Jouve et al., 2001] Jouve, D., Chabbat, B., Amghar, Y., and Pinon, J.-M. (2001). Structuration sémantique de la réglementation. In *Inforsid*, pages 5–25, Martigny, Suisse. 20

[Kahan et al., 2001] Kahan, J., Koivunen, M., Prud'hommeaux, E., and Swick., R. (2001). Annotea : An open rdf. In *Proceedings of the 10th Infrastructure for Shared Web Annotations WS (WWW'01)*, pages 623–632, Hong-Kong. ACM Press. 49

[Kalyanpur et al., 2003] Kalyanpur, A., Hendler, J., Parsia, B., and Golbeck, J. (2003). Smore - semantic markup, ontology, and rdf editor. In *http ://www.mindswap.org/papers/SMORE.pdf*. 49

[Kiryakov et al., 2004] Kiryakov, A., Popov, B., Terziev, I., Manov, D., and Ognyanoff, D. (2004). Semantic annotation, indexing, and retrieval. *J. Web Sem.*, 2(1) :49–79. 47

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. J. (2004). Resource description framework (RDF) : Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210. 48, 53

[Lammel, 2007] Lammel, U. (2007). Business rules make business more flexible. In *3rd Conference on Baltic Business and Socio-Economic Development (BBSED)*. 16, 21, 22

[Lapouchnian, 2005] Lapouchnian, A. (2005). Goal-Oriented Requirements Engineering : An Overview of the Current Research. Technical report, University of Toronto. 31, 32

[Leech, 1997] Leech, G. (1997). Introduction to corpus annotation. In Garside, R., Leech, G., and McEnery, A., editors, *Corpus annotation : Linguistic information from computer text corpora*. Longman 1 : 18. 46

[Legendre et al., 2010] Legendre, V., PetitJean, G., and Lepatre, T. (2010). Gestion des règles « métier ». *Revue Génie Logiciel N°92*. 1

[Lévy et al., 2012] Lévy, F., Guissé, A., and Nazarenko, A. (2012). SBVR as a first step toward the formalization of NL regulations into production rules. In *unpublished*. 41

[Lévy et al., 2010] Lévy, F., Guissé, A., Nazarenko, A., Omrane, N., and Szulman, S. (2010). An Environment for the Joint Management of Written Policies and Business Rules. In Grégoire, E., editor, *Proceedings of the 22nd International Conference on Tools with Artificial Intelligence (ICTAI 2010)*, volume II, pages 142–149, Arras, France. IEE-CPS. FP7. 96

[Lévy et al., 2010] Lévy, F., Nazarenko, A., and Guissé, A. (2010). Annotation, indexation et parcours de documents numériques. *Revue des Sciences et Technologie de l'information (Série Document numérique)*, pages 121–152. 46, 47, 52, 68, 96

[Lovrencic et al., 2006] Lovrencic, S., Rabuzin, K., and Picek, R. (2006). Formal modelling of business rules : What kind of tool to use ? In *Journal of Information and Organizational Sciences*, volume Volume : 30 ; Issue : 2 of *ISSN : 1846-3312*. University o Zagreb, Faculty of Organization and Informatics. 28

[Lukichev and Jarrar, 2009] Lukichev, S. and Jarrar, M. (2009). Graphical notations for rule modeling. A book chapter in Handbook of Research on Emerging Rule-Based Languages and Technologies. Idea Group Inc. 28

[Ma et al., 2009] Ma, Y., Audibert, L., and Nazarenko, A. (2009). Ontologies étendues pour l'annotation sémantique. In *20èmes journées francophones d'Ingénierie des Connaissances (IC09)*, pages 205–216, Hammamet, Tunisie, Tunisie. Quaero, financé par OSEO, agence nationale de valorisation de la recherche. 47, 67

[Ma et al., 2010] Ma, Y., Nazarenko, A., and Audibert, L. (2010). Formal Description of Resources for Ontology-based Semantic Annotation. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC10)*, pages 3765–3772, Malte. 47, 67

[Martínez-Fernández et al., 2008] Martínez-Fernández, J. L., González, J. C., Villena, J., and Martínez, P. (2008). A preliminary approach to the automatic extraction of business rules from unrestricted text in the banking industry. In *Proceedings of the 13th international conference on Natural Language and Information Systems : Applications of Natural Language to Information Systems*, NLDB '08, pages 299–310, Berlin, Heidelberg. Springer-Verlag. 23

[Max, 2006] Max, A. (2006). Writing for language-impaired readers. In Gelbukh, A. F., editor, *CICLing*, volume 3878 of *Lecture Notes in Computer Science*, pages 567–570. Springer. 42

[Miles et al., 2005] Miles, A., Matthews, B., Wilson, M., and Brickley, D. (2005). Skos core : simple knowledge organisation for the web. In *Proceedings of the 2005 international conference on Dublin Core and metadata applications : vocabularies in practice*, DCMI '05, pages 1 :1–1 :9. Dublin Core Metadata Initiative. 52

[Miller and Mukerji, 2003] Miller, J. and Mukerji, J. (2003). MDA guide version 1.0.1. Technical report, Object Management Group (OMG). 23, 29

[Nazarenko et al., 2011] Nazarenko, A., Guissé, A., Lévy, F., Omrane, N., and Szulman, S. (2011). Integrating Written Policies in Business Rule Management Systems. In *Rule-Based reasoning, Programming, and Applications*, volume 6826 of *Lecture Notes in Computer Science*, pages 99–113, Barcelona, Espagne. Springerlink. 96

[Nuseibeh and Easterbrook, 2000] Nuseibeh, B. and Easterbrook, S. (2000). Requirements engineering : a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 35–46, New York, NY, USA. ACM. 32

[OMG, 2008] OMG (2008). Semantics of business vocabulary and business rules (sbvr). Object Management Group (OMG) - Formal Specification, Version 1.0. 19, 26, 36, 38, 40, 41

[Oren et al., 2006] Oren, E., Moller, K. H., Scerri, S., Handschuh, S., and Sintek, M. (2006). What are semantic annotations ? 46, 48

[Paige et al., 2005] Paige, R. F., Kolovos, D. S., and Polack, F. A. C. (2005). Refinement via consistency checking in mda. *Electron. Notes Theor. Comput. Sci.*, 137(2) :151–161. 29

[Popov et al., 2003] Popov, B., Kiryakov, A., Kirilov, A., Manov, D., and Goranov, O. M. (2003). Semantic annotation platform. In *2nd International Semantic Web Conference*, pages 834–849. Springer. 49

[Prié and Garlatti, 2000] Prié, Y. and Garlatti, S. (2000). Sur la piste de l'indexation conceptuelle de documents. *Document Numérique*, 4(1-2) :11–35. 46, 47, 48, 52

[Ralyté, 1999] Ralyté, J. (1999). Reusing scenario based approaches in requirement engineering methods : Crews method base. In *Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, DEXA '99, pages 305–, Washington, DC, USA. IEEE Computer Society. 32

[Ross, 2009a] Ross, R. G. (2009a). *Business Rule Concepts : Getting to the Point of Knowledge (Third Edition)*, volume 157, chapter What Is a Business Rule ? Business Rule Solutions, LLC. 17

[Ross, 2009b] Ross, R. G. (2009b). Rulespeak sentence forms specifying natural-language business rules in english. *Business Rules Journal*, 10(4). 26, 27

[Ross, 2010] Ross, R. G. (October 2010). Five tests for what is a business rule ? *Business Rules Journal*, 11(10). 18

[Schwitter, 2005] Schwitter, R. (2005). A layered controlled natural language for knowledge representation. In *Machine Translation, Controlled Languages and Specialised Languages : Special Issue of Linguisticae Investigationes*, pages 85–106. 29

[Schwitter, 2010] Schwitter, R. (2010). Controlled natural languages for knowledge representation. In *Proceedings of the 23rd International Conference on Computational Linguistics : Posters*, COLING '10, pages 1113–1121, Stroudsburg, PA, USA. Association for Computational Linguistics. 35

[Shaw, 1990] Shaw, M. (1990). Prospects for an engineering discipline of software. *IEEE Softw.*, 7(6) :15–24. 32

[Siddharthan, 2002] Siddharthan, A. (2002). An architecture for a text simplification system. In *Language Engineering Conference 2002 (LEC 2002)*. 42, 43, 44

[Siddharthan and Caius, 2003] Siddharthan, A. and Caius, G. (2003). Syntactic simplification and text cohesion. 42

[Sidhom et al., 2005] Sidhom, S., Robert, C., and David, A. (2005). Analyse automatique de textes comme point de départ d'un processus d'annotation. In *Revue électronique internationale en technologies de l'information (e-TI)*. 46

[Spreeuwenberg and Healy, 2009] Spreeuwenberg, S. and Healy, K. A. (2009). Sbvr's approach to controlled natural language. In Fuchs, N. E., editor, *CNL*, volume 5972 of *Lecture Notes in Computer Science*, pages 155–169, Italy. Springer. 36

[Tauberer, 2008] Tauberer, J. (2008). What is RDF and what it is good for ? `http://www.rdfabout.com/intro`. 53

[Tobon and Franco, 2010] Tobon, H. G. and Franco, A. F. J. (2010). Business rules extraction from business process specifications written in natural language. *Business Rules Journal*, 11(7). 25, 26, 27, 123

[Uren et al., 2006] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., and Ciravegna, F. (2006). Semantic annotation for knowledge management : Requirements and a survey of the state of the art. *Journal of Web Semantics*, 4 :14–28. 46

[Van Lamsweerde, 2001] Van Lamsweerde, A. (2001). Goal-oriented requirements engineering : A guided tour. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, RE '01, pages 249–, Washington, DC, USA. IEEE Computer Society. 31, 32

[Voss et al., 1999] Voss, A., Nakata, K., and Juhnke, M. (1999). Concept indexing. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, GROUP '99, pages 1–10, New York, NY, USA. ACM. 47

[Wagner et al., 2005] Wagner, G., Lukichev, S., Fuchs, N. E., and Spreeuwenberg, S. (February, 2005). First-version controlled english rule language. In *REWERSE IST 506779 Report I1-D2*. 30, 36

[Wan-Kadir and Loucopoulos, 2004] Wan-Kadir, W. and Loucopoulos, P. (2004). Relating evolving business rules to software design. *Journal of Systems Architecture*, 50(7) :367 – 382. 29

[Widlocher and Mathet, 2009] Widlocher, A. and Mathet, Y. (2009). La plate-forme glozz : environnement d'annotation et d'exploration de corpus. In *TALN'09*, Senlis, France. 47

[Zachman, 1987] Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, 26(3) :276–292. 17

# Quatrième partie

# Annexes

# Annexe A

# Code source des requêtes SPARQL

Nous donnons ici en exemple le code source de requêtes SPARQL utilisées dans le module d'accès de données et mises en œuvre dans l'interface de navigation ou l'interface de diagnostic des problèmes de maintenance, ou que nous pouvons utiliser tout simplement pour comprendre notre base de connaissance représentée dans un graphe RDF.

- Liste des concepts de l'ontologie

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?x
where{
?x rdf:type owl:Class.
FILTER regex(str(?x), "http://lipn.univ-paris13.fr/terminae#")
}
```

- Liste de toutes les phrases du corpus

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?x ?y
where{
 ?x rdf:type schema:Sentence.
 ?x schema:content ?y.
}
```

- Liste des phrases contenant le concept "Mileage_credit"

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX onto: <http://lipn.univ-paris13.fr/RCLN/terminae#>

select ?sentence ?content
where
{
 ?sentence rdf:type schema:Sentence.
 ?sentence schema:content ?content.
 optional{
 ?sentence schema:annotatedBy ?textlink.
 ?sentence schema:defineConcept ?concept.
 ?concept schema:realizeConcept <onto:Mileage_credit>.}
```

```
}
ORDER BY ASC(xsd:integer(?sentence))
```

– Liste des concepts apparaissant dans la phrase 12

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX text: <http://lipn.univ-paris13.fr/RCLN/SemEx/text#>

select distinct ?concept
where
{
 ?concept rdf:type schema:Concept.
 optional{
    <text:sent-12> schema:annotatedBy ?link.
    ?link schema:defineResource ??concept.}
}
```

– Liste des concepts présents dans la règle R3

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX rule: <http://lipn.univ-paris13.fr/RCLN/rules#>

select ?concept
where
{
 <rule:R3> rdf:type schema:CandidateRule.
 optional{
 <rule:R3> schema:annotatedBy ?link.
 ?link schema:defineConcept ?concept.}
}
```

– Dans quelles règles occurre le terme "qualifying activity"

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
select ?cr ?content
where
{
 ?cr rdf:type schema:CandidateRule.
 ?cr schema:content ?content.
 optional{
 ?cr schema:annotatedBy ?link.
 ?link schema:defineConcept ?concept.
 ?concept skos:prefLabel <qualifying activity>.
 }
}
```

– Affiche toutes les règles avec leur contexte respectif de transformation dans le processus de normalisation.

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?cr ?content ?versionInitiale ?versionPrecedente ?versionDecompose
where {
  ?cr rdf:type schema:CandidateRule.
  ?cr schema:content ?content.
  optional{
    ?cr schema:originalRuleVersion ?versionInitiale.
    ?cr schema:previousForm ?versionPrecedente.
    ?cr schema:subRule ?versionDecomposee.}
  }
}LIMIT 500
```

– N'affiche que les règles qui sont des révisions

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?revision ?content
where {
  ?rc rdf:type schema:CandidateRule.
  ?cr schema:previousForm ?revision.
  ?revision schema:content ?content.
}LIMIT 500
```

– liste des règles dont l'auteur est demo

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?rc ?content
where {
  ?rc rdf:type schema:CandidateRule.
  ?rc schema:content ?content.
  ?cr schema:author ?rc.
}LIMIT 500
```

– Liste des règles dont un des concepts annotés l'est aussi dans R19

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX rule: <http://lipn.univ-paris13.fr/RCLN/rules#>

select distinct  ?cr ?content
where{
   <rule:R19> schema:annotatedBy ?link.
   ?link schema:defineConcept ?concept.
   optional{
      ?cr rdf:type schema:CandidateRule.
      ?cr schema:annotatedBy ?textlink.
```

```
        ?textlink schema:defineConcept ?concept.
        ?cr schema:content ?content.}
}
```

– Maintenant, les phrases dont un des concepts annotés l'est aussi dans R19

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX rule: <http://lipn.univ-paris13.fr/RCLN/rules#>

select distinct  ?sentence ?content
where{
   <rule:R19> schema:annotatedBy ?link.
   ?link schema:defineConcept ?concept.
   optional{
      ?sentence rdf:type schema:Sentence.
      ?sentence schema:annotatedBy ?textlink.
      ?textlink schema:defineConcept ?concept.
      ?sentence schema:content ?content.}
}
```

– Toutes les règles de 1er niveau qui contiennent atmosphère

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX onto: <http://lipn.univ-paris13.fr/RCLN/terminae#>
PREFIX rule: <http://lipn.univ-paris13.fr/RCLN/rules#>

select distinct ?cr ?content
where {
optional{
    ?rc rdf:type schema:CandidateRule.
    ?rc schema:content ?content.

  optional{
    ?cr schema:annotated ?link.
    ?link schema:defineConcept ?concept.
    ?concept schema:realizeConcept <onto:Atmosphere>.
    ?cr schema:subRule  <R0>.
    ?cr schema:previousForm ?revision.}
    filter(!bound(?revision))
}
```

– Toutes les regles qui sont des revisions où on a ajoute atmosphere

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX onto: <http://lipn.univ-paris13.fr/RCLN/terminae#>
PREFIX rule: <http://lipn.univ-paris13.fr/RCLN/rules#>
```

```
select distinct ?cr ?content
where {
optional{
    ?rc rdf:type schema:CandidateRule.
    ?rc schema:content ?content.

  optional{
    ?cr schema:annotated ?link.
    ?link schema:defineConcept ?concept.
    ?concept schema:realizeConcept <onto:Atmosphere>.
    ?cr schema:previousForm ?prec.}
}
```

– Nombre de règles où apparaîssent le concept "atmosphere"

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX onto: <http://lipn.univ-paris13.fr/RCLN/terminae#>
select count(?rule) as ?ruleNbr
where  {
   ?cr rdf:type schema:CandidateRule.
   ?cr schema:annotatedBy ?link.
   ?link schema:defineConcept ?concept.
   ?concept schema:realizeConcept <onto:Atmosphere>.
}
```

– Toutes les feuilles de règles

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?parent ?content
where {
   ?parent rdf:type schema:CandidateRule.
   ?parent schema:content ?content.
   filter not exists{?fils schema:previousForm ?parent.}
   filter not exists{?fils schema:subRule ?parent.}
}
```

# Annexe B

# Manuel d'utilisation de Semex

EU-IST Integrated Project (IP) 2009-231875 ONTORULE

ONTORULE: ONTOlogies meet business RULEs


## D1.4 Interactive ontology and policy acquisition tools. SemEx User Manual

**Abdoulaye Guissé (Paris 13)**

**with contributions from: Franois Lévy, Nouha Omrane, Adeline Nazarenko and Sylvie Szulman**

**Abstract.**

This document is the user guide of SemEx.
SemEx integrates several components designed to assist business analysts in building a base of business rules out of a policy document. It outputs a set of candidate rules to be used as the specification of a business rule application.
SemEx allows to link textual elements to conceptual resources and to rules. This is helpful to design simplified, self-contained and comprehensive rules, as well as to explore the rule base and trace rule information back to the texts from which it is originated.
The rule editor and annotator integrated in SemEx are inspired of SBVR Structured English although no compliance is checked or required.
Keyword list: Rule edition, Rule normalisation, Rule traceability

| | |
|---|---|
| **Work package number and name** | WP2 Business rules and ontologies ownership and management |
| **Deliverable nature** | Report |
| **Dissemination level** | Public/PU |
| **Contractual date of delivery** | M24 |
| **Actual date of delivery** | June 28, 2011 |

# Executive Summary

This document is the user guide of SemEx.

SemEx is designed to assist business analysts in building a base of business rules out of a policy document, the resulting set of candidate rules being used as the specification of a business rule application.

SemEx allows to link textual elements to conceptual resources and to rules. This is helpful to design simplified, self-contained and comprehensive rules, as well as to explore the rule base and to trace rule information back to the texts from which it is originated.

The rule editor and annotator integrated in SemEx are inspired of SBVR Structured English although no compliance is checked or required.

SemEx has four components and is organized accordingly into four main perspectives: 1/ the annotator inputs a text document together with an ontology and the associated terminology, and outputs annotated text in the rdfa format used by other perspectives ; 2/ the navigator allows to navigate through a document, through an ontology and a rule base and through the links that relate the rules, the concepts, instances and properties of the ontology and their mentions in the document; 3/ the rule editor allows to build a candidate rule base by selecting the relevant text fragments and revising them into simplified, normalized and self-contained statements written in a SBVR-SE style; 4/ the semantic search engine allows to run arbitrarily complex SPARQL queries to explore the semantic space composed of the interlinked ontology, source policy document and rule base for traceability or maintenance purposes.

This documents describes the functionalities of the SemEx platform. The first chapter describes the technical characteristics and the installation instructions. The following chapters present its four main perspectives and their functionalities.



# Table of Contents

# Chapter 1

# Introduction

SemEx is a platform designed to assist business analysts in building a base of business rules out of a policy document, the resulting set of candidate rules being used as the specification of a business rule application.

This platform allows to link textual elements to conceptual resources and to rules. This is helpful to design simplified, self-contained and comprehensive rules, as well as to explore the rule base and trace rule information back to the texts from which it is originated.

The rule editor and annotator integrated in SemEx are inspired of SBVR Structured English although no compliance is checked or required.

SemEx is organized into four main perspectives:

1. The annotator inputs a text document together with an ontology and the associated terminology. It outputs the text annotated with references to the ontology, in the rdfa format used by other perspectives ;
2. The navigator allows to navigate through a document, through an ontology and rule base, and through the links that relate the rules, the concepts, instances and properties of the ontology and their mentions in the document;
3. The rule editor allows to build a candidate rule base by selecting the relevant text fragments and revising them into simplified, normalized and self-contained statements written in a SBVR-SE style;
4. The semantic search engine allows to run arbitrarily complex SPARQL queries to explore the semantic space composed of the interlinked ontology, source policy document and rule base for traceability or maintenance purposes.

SemEx relies on an underlying structure which is a semantic index interlinking textual phrases or fragments, ontological entities and business rules. This index is implemented as an annotated document enriched with HTML links to the ontology and the rule base.

The present document describes the functionalities of the SemEx platform. The first chapter describes the technical characteristics and the installation instructions. The second chapter shows the overall organisation of the platform. The following chapters present its various perspectives and their menus.

## 1.1 SemEx overall interface

The SemEx interface can be described as a main window (see Figure 1.1) consisting of:

- A main menu presenting SemEx characteristics and main actions for loading projects and parameterizing the application (not available);
- The list of opened perspectives. By default only the navigator is opened but new perspectives can be added by clicking on the button `Open perspective` on the left.
- Each perspective encompasses a set of windows and tabs allowing to navigate from one window to another.

## 1.2 Technical Characteristics

- The current version of SemEx platform is compiled using Sun 1.6 Java virtual machine.
- It relies on UTF8 text encoding.
- It had been designed for English[1].

## 1.3 Installation

To install SemEx, you need java, version 1.6.

Download the SemEx application zip file corresponding to your operating system:

- `linux_64.zip` or `linux.gtk.x86.zip` for Linux;
- `macosx.cocoa.x86_64.zip` for Mac OS, 64 bits;

[1]Although portability to other languages, such as French, is not scheduled yet, it is quite simple. It requires a terminology in that language for the working ontology and possibly an adaptation of the annotator to different word ordering rules. All other components are language independent.



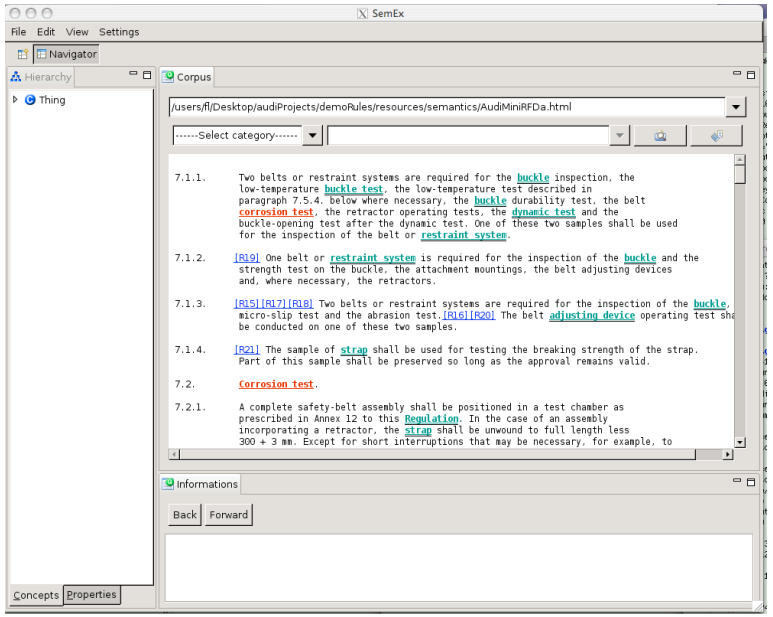



Figure 1.1: SemEx overall interface

- `win32.win32.x86.zip` for Windows.

Then decompress the above file and start the SemEx tool as your operating system ordinarily does for Eclipse applications. To experiment the tool on existing data, you can also load and decompress two files providing three sample projects:

- `creatingProjects.zip` contains sample data to use the annotator or import a project,
- `existingProjects.zip` holds the demoRule and demoQuery projects.

## 1.4 Running SemEx

You will generally be working on an already annotated document and create or reopen the corresponding project (one project may hold several documents). When SemEx starts,

a dialog window appears (see Figure 1.2). You must type in the name of your project and select the parent directory where the project directory will be located in, using the `browse` button (opening the dialog in figure 1.3). Then, click the (`Load data` button. If you entered the name and location directory of an existing project (e.g. `demoRules` and `existingProjects`), you will be asked to confirm (Figure 1.4), and then offered the possibility to change the data (Figure 1.5). If you don't, the project opens.

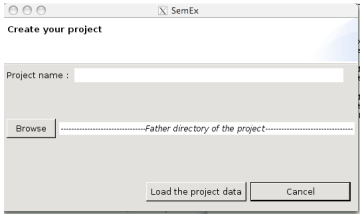



Figure 1.2: Dialog window: creation of a project

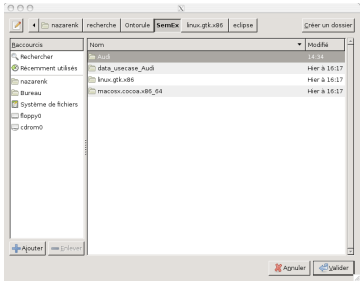


Figure 1.3: Dialog window: navigation through the file system

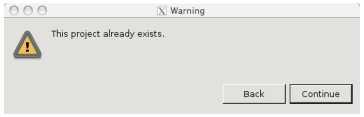



Figure 1.4: Warning window: loading an existing project

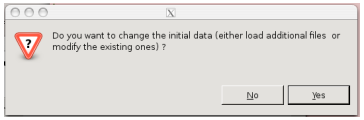



Figure 1.5: Dialog window: changing the data of an existing project ?

In the case you are creating a new project or you have decided to change the data of an existing one, you are offered to choose the sources. You will get a new dialog window to load semantic and linguistic information (Figure 1.6):

**Semantic information** You need at least three files: the RDFa annotations, the document in a special xml format and the ontology. If you also want to import existing



rules, load `rules.rdf` and overide the existing one. Use the `Browse` button to identify the relevant directory and select the required files. If you select the `creatingProjects/importing` directory, you will find the `semantics` directory, with the following files :

- `MiniRegulationRFDa.html`
- `corpus2XML.rdf`
- `ontology.owl`

Select them all (Figure 1.7) and you will get the window represented in Figure 1.8.

**Linguistic information** It is in the `SKOS` file. Use the `Browse` button to identify the relevant directory and select the required file. If you select the `creatingProjects/importing` directory, you will find the `skos` directory with a `skos.rdf` file. It you select that file (Figure 1.9), you will get the the window represented on Figure 1.10.

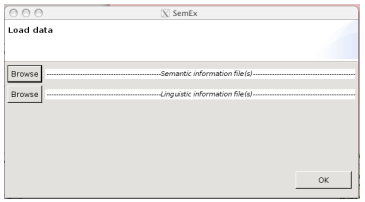



Figure 1.6: Dialog window: loading data

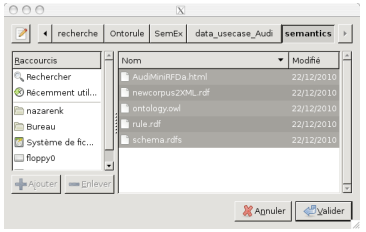


Figure 1.7: Dialog window: loading semantic information

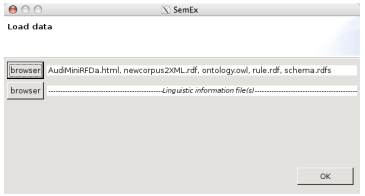



Figure 1.8: Dialog window: semantic data loaded

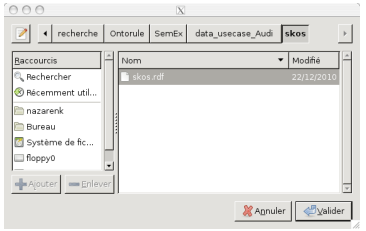


Figure 1.9: Dialog window: loading linguistic information

Last, in the case you are creating a new project and, in the loading data dialog window (Figure 1.6), you do not choose any source data, SemEx guesses you need first to create an annotated text and enters straight the annotation perspective (Figure 1.12.

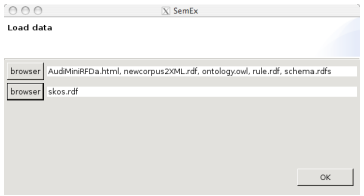



Figure 1.10: Dialog window: semantic and linguistic data loaded

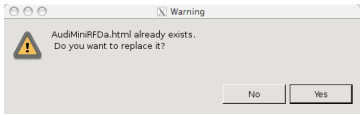



Figure 1.11: Warning before erasing existing data

Note that if you open an existing project and choose to load additional data, the technique is fairly simple but a bit rough: if you load a file with the same name as an existing data file, the new data will erase the previous ones. You are warned for each replaced file (see Figure 1.11).

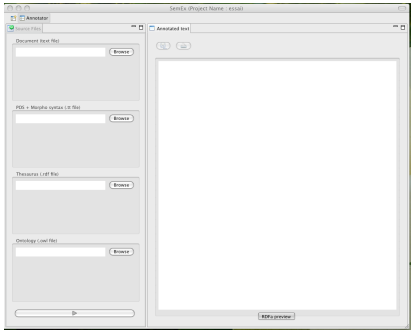


Figure 1.12: The annotation perspective

Once this is done, the project and relevant data are loaded and you are ready to work (see Chapter 2 for a general presentation):

- to annotate documents (see Chapter 3),
- to edit rules (see Chapter 5),
- to navigate through the semantic and document bases (see Chapter 4),
- and to perform some specific semantic searches (see Chapter 6).

# Chapter 2

# General presentation of SemEx

SemEx is a Semantic Explorer tool that helps a business analyst to edit, explore and update a set of business rules and more generally a business model anchored in policy documents for a given application.

## 2.1 Inputs/outputs

SemEx takes as input:

1. A document that is used as the initial source of information for building the business rules (plain text format, see Figure 2.1);
2. A domain specific ontology (OWL format);
3. A linguistic knowledge base that makes explicit the preferred and alternative terms that are associated with the concepts, instances and properties of the ontology (SKOS format, see Figure 2.2).

```
7.1.1.      Two belts or restraint systems are required for the buckle inspection, the
            low-temperature buckle test, the low-temperature test described in
            paragraph 7.5.4. below where necessary, the buckle durability test, the belt
            corrosion test, the retractor operating tests, the dynamic test and the
            buckle-opening test after the dynamic test. One of these two samples shall be used
            for the inspection of the belt or restraint system.

7.1.2.      One belt or restraint system is required for the inspection of the buckle and the
            strength test on the buckle, the attachment mountings, the belt adjusting devices
            and, where necessary, the retractors.
```

Figure 2.1: Example of source document

SemEx also uses the results of a morpho-syntactic analysis of the document. As this analyser is an external tool, in the current version its results must be provided in a separate file.

```
– <rdf:RDF>
  – <rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#Conditionning">
      <skos:prefLabel>conditionning</skos:prefLabel>
      <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#CorrisionTest">
      <skos:prefLabel>corrision test</skos:prefLabel>
      <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#Time">
      <skos:prefLabel>time</skos:prefLabel>
      <skos:altLabel>hour</skos:altLabel>
      <skos:altLabel>minute</skos:altLabel>
      <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#TestOfBreakingStengthOfStrip">
      <skos:prefLabel>breaking strength test</skos:prefLabel>
      <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#LightConditionning">
      <skos:prefLabel>light conditionning</skos:prefLabel>
      <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    </rdf:Description>
  – <rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#LowTemperatureChamber">
      <skos:prefLabel>low temperature chamber</skos:prefLabel>
      <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    </rdf:Description>
```


Figure 2.2: Example of SKOS input

SemEx outputs:

1. The input document annotated with respect to the semantic ontological elements and rules (RDFa format, see Figure 2.3);
2. The list of rules that have been created by the business analyst (RDF format, see Figure 2.4);
3. Possibly, an update of the input ontology (OWL format, not available);
4. Possibly, an update of the input linguistic information (SKOS format, not available).

## 2.2 Main functionalities

SemEx offers four main functionalities, each one corresponding to a specific eclipse perspective:



7.2.2. On completion of the exposure test the assembly shall be gently washed, or dipped in clean running water with a temperature not higher than 38 C to remove any salt deposit that may have formed and then allowed to dry at room temperature for 24 hours before inspection in accordance with paragraph 6.2.1.2. above.

7.3. Micro-slip test (see Annex 11, figure 3 to this Regulation).

7.3.1. The samples to be submitted to the micro-slip test shall be kept for a minimum of 24 hours in an atmosphere having a temperature of 20 + 5 C and a relative humidity of 65 + 5 per cent.[R2] The test shall be carried out at a temperature between 15 and 30 C.

Figure 2.3: Example of RDFa annotation. The colored text corresponds to active HTML links. The annotated terms refer to ontological entities and the bracketed text refers to rules.

**Annotator** This perspective enables to annotate a text, either a large document or a small fragment of text, with respect to a given ontology. The result is a navigable RDFa document (Figure 2.3) where the annotated elements are HTML link anchors.

**Rule editor** This perspective enables the business analyst to select the text fragments that are relevant for the business rule application to develop, edit them and step by step turn them into normalized candidate rules.

A candidate rule is a (seem-)controlled language statement that expresses a rule or constraint to take into account in the aimed rule application. The controlled language is close to SBVR Structured English (SBVR SE), although neither the conformance with SBVR SE nor the syntactic validation is required. The business analyst is free to go as further as he/she wants in this normalization process. Compared to the initial text fragments, the resulting candidate rules are supposed to be more self-contained and explicit, lexically and syntactically normalized, less redundant and more straightforward.

**Navigator** This perspective enables the business analyst to navigate in the semantic space composed of the ontology and the text. The hierarchies of concepts and properties that form the ontology can be explored, the text can be browsed and the links between the ontology and the text can be exploited to locate the occurrences of one concept in the text or to identify the information of the concept that is related to a given term occurrence in the text.

**Search engine** This fourth perspective enables the business analyst to explore the semantic space formed by the text, the ontology and the rule base which are linked to each other. The user can type in a SPARQL query, run it and visualize the list of results: a mix of sentences, ontological elements and rules.

```
- <rdf:RDF xml:base="http://lipn.univ-paris13.fr/RCLN/ontorule/Audi/rules">
    <Rule rdf:ID="R0"/>
  - <Rule rdf:ID="R2">
    - <ruleText>
        <span rel="schema:annoted" typeof="schema:Rule"><span about="textlinkR2" rel="schema:defineResource"
        typeof="schema:TextLink"><span id="if" type="Keyword" class="Keyword">If</span> the <span
        rel="schema:realizeConcept" typeof="schema:Concept"><a about="http://lipn.univ-paris13.fr/RCLN/terminae
        /Audi#Method" class="Concept" href="#" onclick="function1('Method');" target="_blank">method</a></span> is <span
        id="not" type="Keyword" class="Keyword">not</span> carried out immediately after <span rel="schema:realizeConcept"
        typeof="schema:Concept"><a about="http://lipn.univ-paris13.fr/RCLN/terminae/Audi#Conditionning" class="Concept"
        href="#" onclick="function1('Conditionning');" target="_blank">conditionning</a></span>, the specimen <span id="shall"
        type="Keyword" class="Keyword">shall</span> <span id="be" type="Keyword" class="Keyword">be</span> placed in a
        hermetically-closed receptacle <span id="until" type="Keyword" class="Keyword">until</span> the <span
        rel="schema:realizeConcept" typeof="schema:Concept"><a about="http://lipn.univ-paris13.fr/RCLN/terminae
        /Audi#Method" class="Concept" href="#" onclick="function1('Method');" target="_blank">method</a></span> begins.
        </span></span>
      </ruleText>
      <date>03/01/2011</date>
      <author/>
      <pattern/>
      <type/>
      <hasPremise/>
      <hasConclusion/>
      <subRule rdf:resource="#R0"/>
    </Rule>
  </rdf:RDF>
```


Figure 2.4: Example of rule list (R2 is a son of the root R0) as is output by the SemEx platform

## 2.3 Opening and switching perspectives

When SemEx is launched, a single perspective is opened: either the default navigation perspective (see Figure 1.1), or the annotation perspective (Figure 3.1 (if a new project is created without providing an annotated document); but you can open the other perspective by clicking on the left 'plus' button of the perspective menu (see Figures 2.5 and 2.6) and by selecting the perspective you want to open.

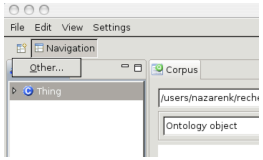



Figure 2.5: New perspective button in the perspective menu

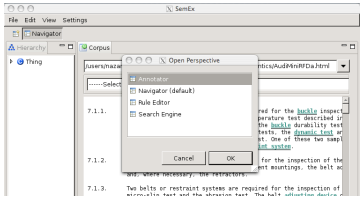



Figure 2.6: Opening a new perspective

# Chapter 3

# Annotation

## 3.1 Presentation of the perspective

The annotation perspective (Figure 3.1) is composed of two main zones.

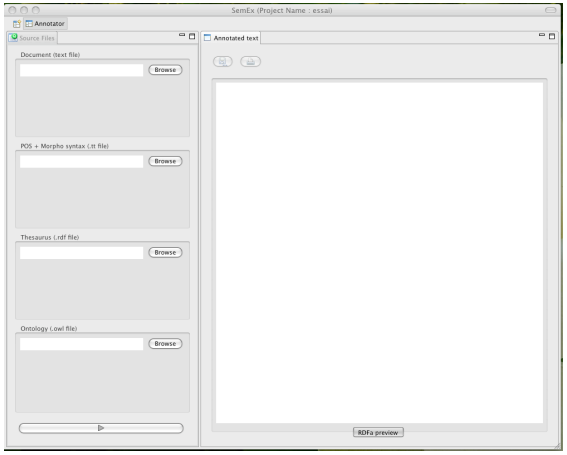



Figure 3.1: The annotation perspective

- In the left panel, four textual fields are used to choose the input, Down this panel is a button to run the annotator.

- In the right pannel, a large text zone allows to visualize the annotations. Two buttons at the top of the panel allow saving the result or printing it.

## 3.2 Functionalities

The perspective has a single functionality: annotating a plain text with respect to a given ontology. At first, the user provides the name of input files (Figure 3.2):

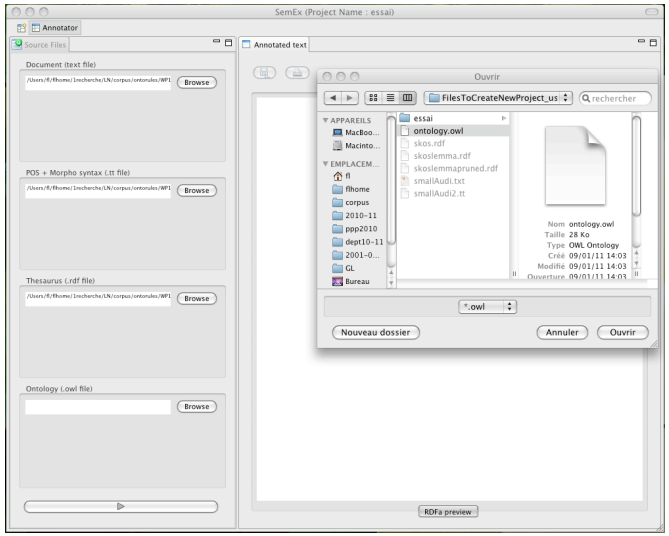



Figure 3.2: Annotator's input files

- the *document file* (`.txt`) contains the text to be annotated;
- the *ontology file* (`.owl`) contains the entities used for the annotation;
- the *thesaurus file* (`skos.rdf`) contains linguistic information linking entities to possible textual occurences;
- the *part-of-speech (POS) information file* (`.tt`) associates to each word its lemma (lexical entry) and syntactical category.



The document is plain text, so as to be annotated in RDFa with HTML tags. Converters exist to transform PDF or Word files while preserving the presentation. The ontology and the thesaurus are output by TERMINAE. The former is an OWL file, the latter follows the SKOS norm – see the TELIX proposal in annex A of Ontorule D1.4 deliverable for possible evolutions. You need to use an ontology and a thesaurus which are inline – the thesaurus describes the linguistic features of this ontology. You can apply them to any text, but if the text's domain or genre is very different of TERMINAE's model, the result may be poor.

The part-of-speech information file is produced from the document by a tagger. TERMINAE also relies on a tagger to create the thesaurus, and, for consistency, the same one must be used along the chain. Presently, TreeTagger is used. It is free software available from `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`. To avoid relying on a pre-existing installation of TreeTagger on all the users machines, the `.tt` file that it outputs is input separately into SemEx. It is thus possible to experiment SemEx using pre-computed `.tt` files. In particular, `.tt` files are provided with the example documents, with the same base name as the document itself.

Once indicated the names of input files, clicking the "Run" button shows a preview of the annotated text in the right panel (Figure 3.3):

You need then to save the annotated document (at the top of the right pannel). You are offered (Figure 3.4) to add it to the current project, or to choose another location.

**Example**: On the same page as SemEx has been downloaded, you find a ziped directory "creatingProjects". Download and unpack it: the subdirectory "usingAnnotator" contains the four relevant files. Now run SemEx, create a new project and provide no source data. In the annotation perspective which opens, the *Browse* search of each text field is aware of the relevant type, so you only need to reach the "usingAnnotator" directory. Once all the fields are informed, click the button at the bottom to run the annotator. Then use the "save" button at the top of the preview panel and save the annotations in the current project. You are ready to navigate and acquire rules. ◁

Note that the intention of a project is to gather documents which share the same ontology. After the first document is annotated, it is not good practice to annotate a new one with a different .owl or .skos file and to save it in the project. In this case, it is safe to save the annotations at a different location - e.g. in the *semantic* subdirectory of the project where the owl and skos file come from.

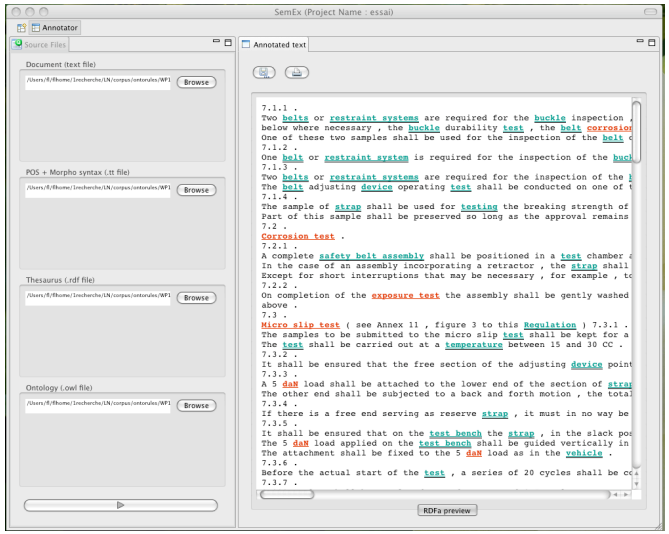



Figure 3.3: Visualizing the annotations

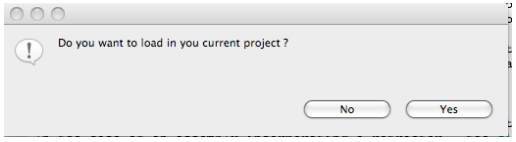



Figure 3.4: Saving the annotated document

# Chapter 4

# Navigation

This chapter gives a general presentation of the navigation perspective and illustrates the benefit of the navigation functionalities on few examples. The sample data used in this chapter may be loaded either by using any of the samples in `creatingProjects` with the related method, or by opening demoRule in `existingProjects`

## 4.1 Presentation of the perspective

The navigation perpective is composed by default of three main windows (see Figure 4.1) but you can modify this presentation using the minimize or maximize buttons associated to each window (see Figure 4.2) :

**Hierarchy window** The ontological hierarchy is presented on the left of the perspective. Two tabs at the bottom of the window allow to visualize either the concept hierarchy (Figure 4.3) or the property hierarchy (Figure 4.4). Clicking on the black triangle to the left of a concept or property allows to visualize or hide the sub-hierarchy of that concept or property.

**Corpus window** The corpus is presented on the top right part of the navigation perspective (see Figure 4.5). This window is itself composed of three areas:

- The top area is a field where the name of the corpus file appears. You can switch from one file to the other if several files have been loaded.
- The bottom area presents the annotated corpus in which the terms that are annotated with ontological entities are emphasized and the rules are annotated. The colors indicate which type of annotation is attached to a given term (green for concept annotations, red for concept instance annotations, blue for property annotations[1]). The rules are noted in brackets

[1]The property annotations are are not available yet.

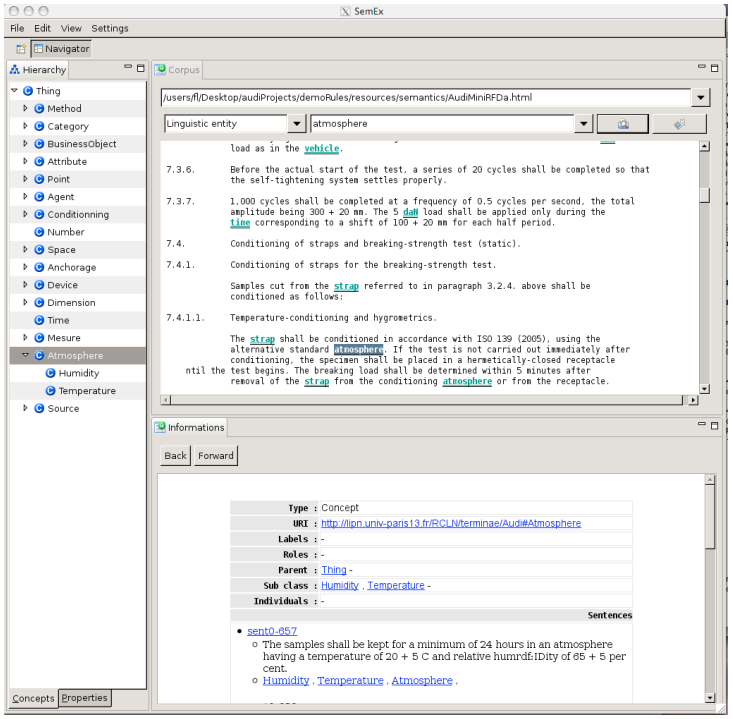



Figure 4.1: Navigation perspective

- The middle area offers search functionalities. The left field allows to select the type of information you are looking for: linguistic entity, candidate rule, ontological entity (see Figure 4.6). The middle field allows you to select the element you are interested in (the linguistic entity `vehicle` in the example of Figure 4.7). The right button (Figure 4.8) allows to locate the various occurrences of the selected element[2].

**Information window** The right bottom part of the navigation perspective is used to visualize the information attached to a given ontological entity, as the concept instance `PointA` in the example of Figure 4.1 or the concept `Regulation` in Figure 4.9.

[2]Only forward search is currently available.



Figure 4.2: Minimize and maximize buttons

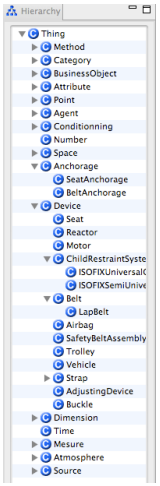



Figure 4.3: Concept hierarchy

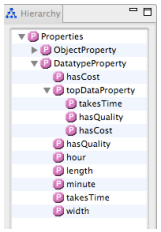



Figure 4.4: Property hierarchy

## 4.2 Functionalities

The navigation perspective enables to navigate through the underlying index which is a semantic space composed of the interlinked corpus, ontology and rule base. Various operation are supported:

**Locating occurrences** Selecting any ontological entity in the hierarchy area initializes the search field automatically with the selected entity and enables to search for related annotated occurrences in the text, which appears in the corpus area.

For instance, selecting the concept `Vehicle` enables to locate either the `vehicle` term occurrences (selecting the category `Linguistic entity` attached to the concept `Vehicle`) or the pieces of texts directly annotated with that concept (selecting the category `Ontological entity`). This enables to locate the occurrences of a concept whatever linguistic form appears in the text (*seats* as well as *seat* for the `Seat` concept, for instance) or the occurrences annotated with a subclass of the selected concept (`Hour` as well as `Time`, for instance).

**Analyzing detailed information** The linguistic information area allows to look at the detailed information attached to a given ontological entity.

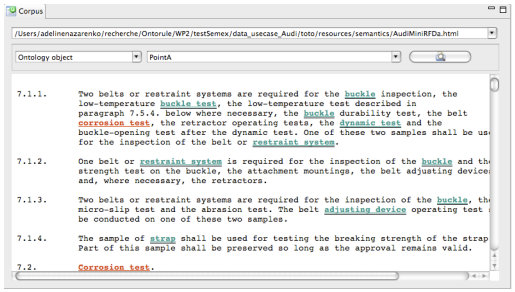



Figure 4.5: Corpus window

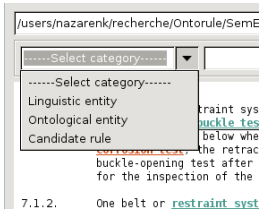



Figure 4.6: Selecting an information category

For instance, the figure 4.9 shows that the concept `Regulation` is associated with an `URI`, a `Parent` concept (`Document`), a `Sub class` and `Individuals` (or instances). Moreover, the list of the sentences where the selected concept appears as an annotation is shown at the bottom of the information window[3]. The number of these sentences, as such, is already an indication of the role of the ontological entity in the text.

**Exploring the semantic index** Following the HTML links enables to explore the semantic space that form the interlinked corpus, ontology and rule base. Clicking on an ontological entity in the hierarchy window updates the information fields, and gives a look to the list of the sentences that refer to that ontological entity. Pointing on any sentence id shows the text of that sentence and the list of co-occurring annotated entities. In future the same will apply for the rules in which the selected ontological entity appears, but this functionality is not available yet. Clicking on any active

[3]The list of rules where the concept is referred should also appear but this functionality is not available yet.



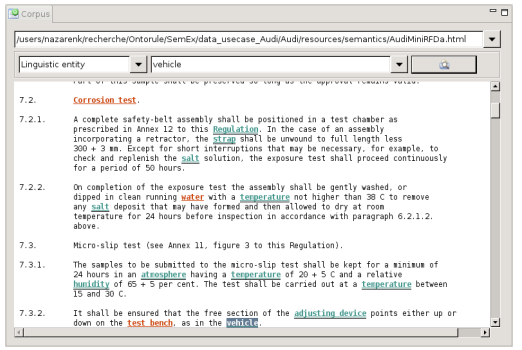



Figure 4.7: Selecting an information category: example of *vehicle*

Figure 4.8: Button for searching occurrences

entity in the information window updates the window for that new entity.

These various functionalities make the navigator a powerful device for exploring the semantics of a corpus and the associated business model.

Figure 4.9: Information window

# Chapter 5

# Rule editing

This chapter gives a general presentation of the rule editor perspective and illustrates the benefit of the editing functionalities on some examples.

## 5.1 Presentation of the perspective

### 5.1.1 Overview

The rule editing perpective is composed by default of three main windows (see Figure 5.1) but you can modify this presentation using the minimize or maximize buttons associated to each window (see Figure 4.2):

**Hierarchy window** A hierarchical view of the rule base is presented on the left of the perspective. If you open a new project, the initial rule base is empty except for a root rule, `R0`. When new rules are created, the hierarchy of rules is enriched as shown on Figure 5.2. Clicking of the black triangle on the left of a rule allows to visualize or hide the sub-hierarchy of that rule.

**Rule window** The window allowing to edit rules, to visualize them in details and to browse sets of rules is presented in the top right part of the rule editor perspective. It should be composed of three alternative views, although only the two first ones are available so far (see Figures 5.3 and 5.4). As usual, the tabs allow to switch from one view to the other. These views are presented in Section 5.1.3.

**Information window** The right bottom part of the rule editor perspective is used to visualize the information attached to a given ontological entity, as in the navigation perspective (p. 16 and 19).





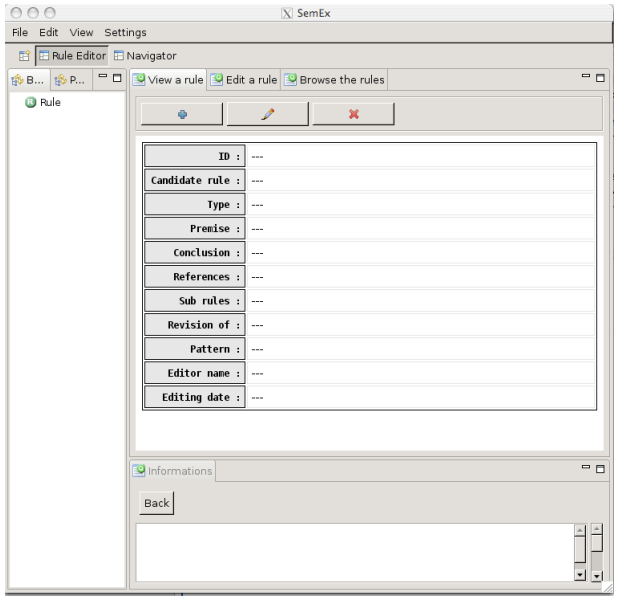



Figure 5.1: Rule editor perspective

### 5.1.2 Presentation of the rule base

The rule base is structured along two complementary hierarchical axes:

1. A rule $R_b$ is a revision of a rule $R_a$ if $R_b$ has been created through the transformation of $R_a$ (see following section).
2. A rule $R_c$ is a sub-rule of $R_a$ is $R_a$ has been decomposed into several rules, one of them being $R_c$.

In the hierarchical view of the rule base presented on Figure 5.2, both relations are represented. A daughter rule represented in red is a revision of its mother rule. A daughter rule represented in green is a sub-rule of its mother rule.

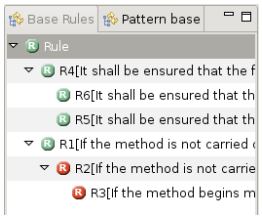



Figure 5.2: Rule base window

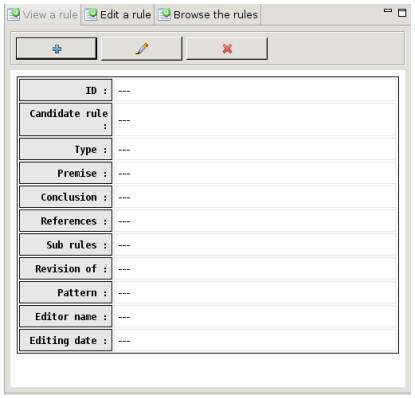



Figure 5.3: `View a rule` view

### 5.1.3 The rule window

The rule window is composed of three different views, but the third one is not available yet.

#### View a rule

The first view (`View a rule`) presents rules in details. It is itself composed of two areas:

**Menu** The menu presents three buttons (see Figure 5.5). From left to right:



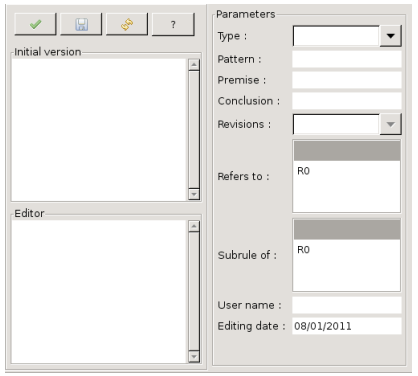



Figure 5.4: `Edit a rule` view

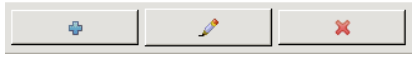


Figure 5.5: Menu of the `View a rule` view

**New candidate rule** Click on that button to add a candidate rule to the rule base. You will get a dialog window (see Figure 5.6, and Section 5.2 p. 25).

**Edit a candidate rule** Click on that button after selecting an existing rule to edit it. This opens the `Edit a rule` view, initialized with the selected rule.

**Delete a candidate rule** Click on that button to delete the selected candidate rule. This updates the hierarchical view of the rule base.

**Rule description** The rule description area is composed of 11 informative fields: the rule `ID`, its statement (`Candidate rule`), its `Type`, `Premise` and `Conclusion` if they are known, some `References`, the IDs of the rules from which it is derived either by decomposition (`Sub rule`) or by revision (`Revision of`), the possible rule pattern to which is corresponds (`Pattern`), the `Editor name` and the `Editing date` (see Figure 5.3).

#### Edit a rule

The second view of the rule window (`Edit a rule`) allows to edit a rule (see Figure 5.4). It is itself composed of two areas:

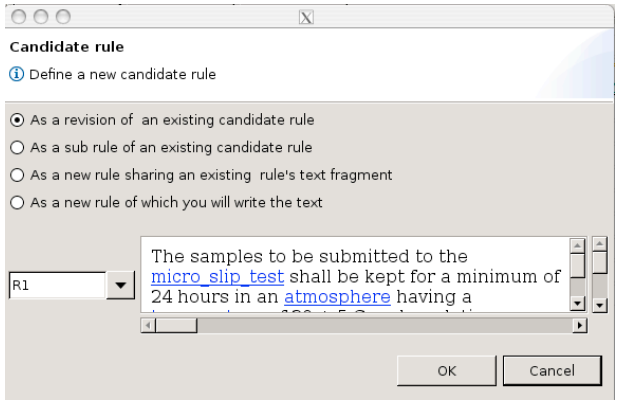



Figure 5.6: Dialog window: defining a new candidate rule

- The left area allows to visualize the rule to transform and to edit it. It is organized vertically into three parts:

  **Menu** The menu is composed of 4 buttons (see Figure 5.7):

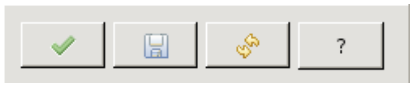



Figure 5.7: Menu of the `Edit a rule` view

  **Validate** Click on that button to validate the rule (this functionality is not available yet).

  **Save** Click on that button to save the new rule in the rule base.

  **Refresh** Click on that button to reinitialize the view.

  **Help** Leaving the pointer on that button reminds you of the role of the function keys: `F1`, `F2` and `F3` respectively give access to the lists of the buffer words, of the keywords and the terms; they can be used while editing a rule.

  **Initial Version** This field is initialized by the rule to be revised if the new rule has been created as a revision of a preexisting rule. It is empty otherwise.

  **Editor** In this field, you can edit the new rule. You can copy, paste and transform a source rule or write it from scratch. The words belonging to the keyword list



or to the conceptual vocabulary associated to the ontology are automatically colored.

- The right area presents the characteristics of the edited rule as in the `View a rule` view of the rule window.

**Browse the rules**

The third view of the rule window (`Browse the rules`) will allow to explore the rule base according to pattern-based or string-based criteria but it is not available yet.

## 5.2 Rule editing

The rule editor perspective is used to edit new candidate rules or to revise existing ones. The general process of editing rules relies on the following operations:

**Text browsing** If you want to extract rules from a source document, you first browse the text looking for the fragments that express relevant rule information for the rule application you are designing. This selection can be made:

- By browsing the text and manually selecting a text fragment worth turning into a candidate rule;
- By applying rule extraction patterns on the corpus and analyzing the returned fragments (This functionality is not available yet).

**Creation of a new rule** You can create a rule from scratch or from a selected text fragment:

- If you have selected a text fragment, you can turn it automatically into a candidate rule by clicking on the right button of your mouse. After confirmation (see Figure 5.8), this opens the `Edit a rule` view in which the `Initial version` and `Editor` fields are initialized with the text you have selected. Note that the source text fragment is then annotated with the rule you are creating (the name of the rule appears in brackets).
- If you want to create a candidate rule from scratch, click on the `New candidate rule` button in the menu of the `View a rule` view. In the `Edit a rule` view that appears, the fields are empty but you can directly type in the new candidate rule in the `Editor` field.

**Revision of an existing candidate rule** You can create a new candidate rule as a revision of an existing rule, which may be a text fragment or an output of a previous revision process. In that case, the initial version appears in the top and bottom fields. You

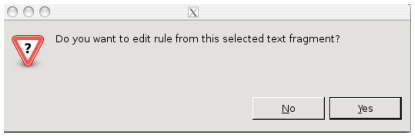



Figure 5.8: Warning window: creating a candidate rule out of a selected text fragment

can edit it in the bottom field and revise it. Once it is saved, the resulting rule appears in red in the hierarchical view of the rule base.

**Creation of a sub rule of an existing rule** You can create a new candidate rule as a sub-rule of an existing candidate rule, which means that you decompose the initial rule into various sub-rules. The same editing process applies in the editor field. Once it is saved, the resulting rule appears in green in the hierarchical view of the rule base.

You have access to these various options by clicking on the `Add candidate rule button` in the menu presented on Figure 5.5). This open the dialog window presented in Figure 5.6.

When editing a rule, various word lists are accessible for automatic completion and for the automatic annotation of the text you type in[1]:

- The buffer list consists of all the words that have been typed in the same session;
- The keyword list consists of all the keywords of SBVR SE;
- The term list consists of the conceptual vocabulary attached to the loaded ontology.

The annotated fragments of texts are highlighted with colors as for the corpus in the navigator perspective:

- The keywords are colored in orange.
- The color of a term depends on its type:
  - It is green if it refers to a concept,
  - It is red if it refers to a concept instance,
  - It is blue if it refers to a property.

Some of the parameter fields are automatically updated when you create a rule (see Figure 5.9). Others have to be filled manually. An automatic analysis of the rules might help to fill these remaining fields automatically but this functionality is not available yet.

[1]This information is stored in the input SKOS file.



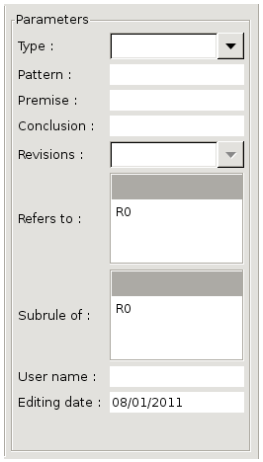



Figure 5.9: Rule parameters

## 5.3 Transformation process

As a principle to classify the transformations performed to clarify a candidate rule, a *revision* proposes a single replacement candidate rule which has the same meaning ; a *sub-rule* do not carry over the full meaning of its parent candidate rule, so several sub-rules are needed for that. Various operations can lead to rewrite a candidate rule:

**Lexical normalisation** The simplest revision consists in substituting plain words by key words and by the preferred terms associated with the ontology. This leads to normalize the rules. In the example presented on Figure 5.10, the word *test* has been substituted by the term *method*, as `Method` is the generic concept that encompasses all specific tests (as shown on the partial ontology presented on Figure 5.11), and *test* is one of its labels, not the prefered one, in the SKOS file.

**Syntactic transformation** A more complex revision consists in transforming the syntax of the rule to get a simplified and more straightforward wording. Various transformation can me made, such as:

- Making explicit the premise and conclusion of the rule. For instance, the structure of the sentence in Figure 5.12) is clarified by expliciting a *then* at the place where consequence was marked by a conjunction;

- Eliminating negations in the premise when they can have a positive counterpart: in Figure 5.13, the phrase "not carried out immediately after the conditioning" is transformed into "begins one minute after conditioning", which expresses a condition that is easier to check. Note that the new formulation here adds to the initial one. It need being more precise to decide when the rule applies, and this transformation has to controled by business people.
- Transforming verbal forms into nominal ones (or conversely) to stick to a standard formulation. For instance, "the strap breaks at a load of ..." is transformed into "the breaking load of the strap is ...".

**Splitting** A candidate rule may be subdivided into several simpler ones. A trivial case is the rule "The temperature must be maintained between 20 and 28 degrees" which yields "The temperature must be maintained over 20 degrees" and "The temperature must be maintained under 28 degrees". More complex cases can be found in the examples provided by Ontorule.

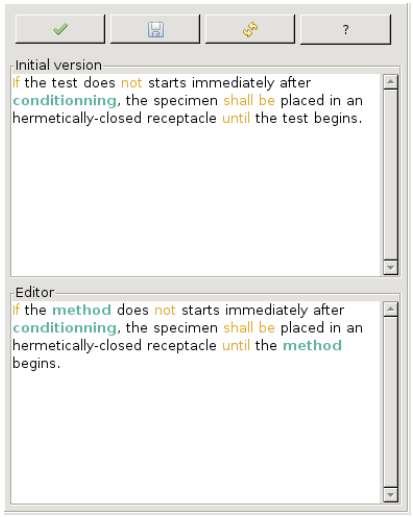



Figure 5.10: Example of lexical normalisation. The bottom field presents a revised version of the above statement.

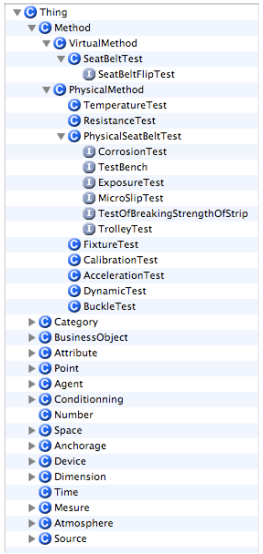



Figure 5.11: Sub hierarchy of the `Method` concept



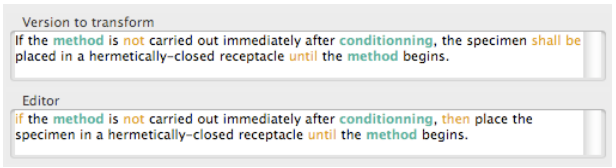



Figure 5.12: Example of making the structure explicit. The bottom field presents a revised version of the above statement.

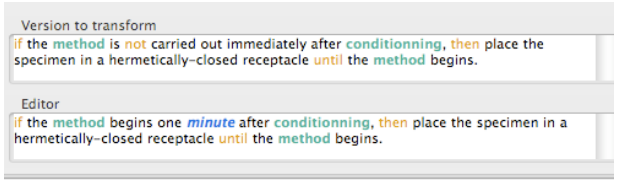



Figure 5.13: Example of eliminating negation. The bottom field presents a revised version of the above statement.

Other transformations can lead to eliminate redundancies in the candidate rules, make explicit presuppositions or implicit statements. But it is up to the business analyst to choose when to stop the revision process (see for instance, the final result of the above transformation on Figure 5.14).

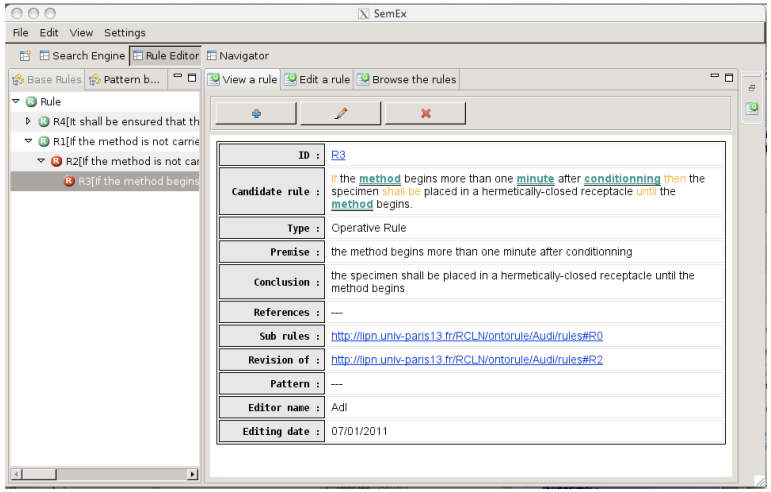



Figure 5.14: Result of the rule revision process. As shown on the hierarchy on the left the candidate rule $R3$, which texts and characteristics are described on the right is a revision of $R2$, which is itself a revision of $R1$, which is a sub-rule of $R0$, as any rule.

# Chapter 6

# Semantic search

This chapter gives a general presentation of the search engine perspective and illustrates the benefit of the search functionalities on a simple example.

## 6.1 Presentation of the perspective

The search engine perspective is composed a single window (see Figure 6.1), which is itself composed of two main fields:

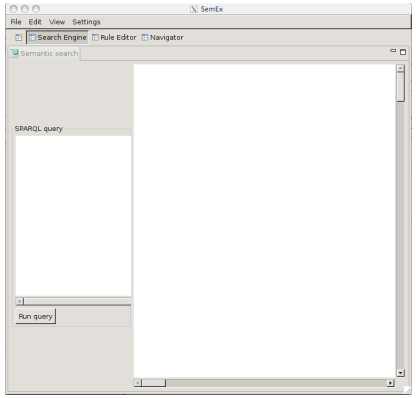



Figure 6.1: Search engine perspective

**SPARQL query** The first field, located on the left, allows to type in a SPARQL query;

**Results** The second field, located on the right, presents the result of the query.

In addition to that a `Run query` button allows you to run the query that appears in the `SPARQL query` field on the semantic space composed of the interlinked corpus, ontology and rule base.

Using the search engine requires that you know SPARQL syntax. Some queries are predefined and can be directly accessed through the graphical user interface, and more can be made easily accessible as they reval interesting. But any query which is not available yet as a predefined one can be made through this perspective.

## 6.2 Writing requests

A simple request to list the rules is :

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
select ?x ?text ?type ?date ?author
where {
  ?x rdf:type schema:Rule.
  ?x schema:ruleText ?text.
  ?x schema:type ?type.
  ?x schema:date ?date.
  ?x schema:author ?author.
}LIMIT 500
```

The PREFIX keyword is only a macro for abbreviation purposes. The file schema.rdfs defines SemEx specific relations : ruleText, type, date, author. This file is in the subdirectory `semantics` of the project you have opened. It defines also the rdf: prefix. The select and where clauses are easy to understand. The full syntax is described in the SPARQL specification. The result is in figure 6.2.

## 6.3 Functionalities

The search engine perspective allows to navigate through the semantic space composed of the interlinked corpus, ontology and rule base. You can, for instance, look for:

- All the rules that have a given concept $C$ in the premise and a given keyword $k$ in the conclusion;
- All the sentences which are annotated by several rules or which are annotated by two disjoint concepts;



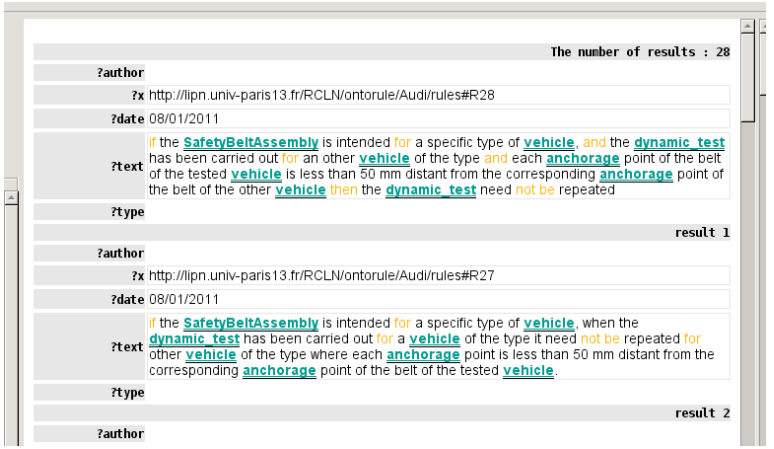



Figure 6.2: Result of a query

- The ontological entities used as annotation in more than $n$ sentences;
- The set of rules sharing more than 4 ontological entities.
- etc.

For instance, if you run the query presented in Figure 6.3, you will get all the rules that share at least one concept with a given rule $R19$ (see Figure 6.4).

```
PREFIX schema: <http://lipn.univ-paris13.fr/RCLN/schema#>
PREFIX audirules: <http://lipn.univ-paris13.fr/RCLN/ontorule/Audi/rules#>
select distinct   ?rule ?content
where{
  audirules:R19 schema:annoted ?link.
  ?link schema:defineResource ?resource.
  ?resource schema:realizeConcept ?concept.
  optional{
    ?rule schema:ruleText ?content.
    ?rule schema:annoted ?textlink.
    ?textlink schema:defineResource ?source.
    ?source schema:realizeConcept ?concept.
  }
}
ORDER BY ?rule
```

Figure 6.3: Example: querying for rules (both rule content and rule ID) that share at least one concept with the rule $R19$

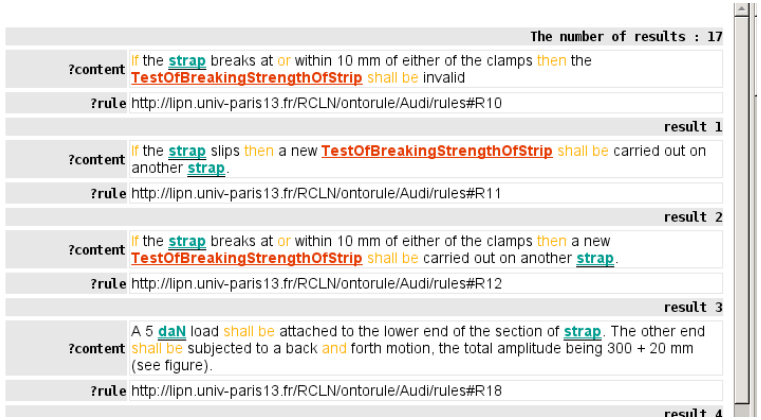



Figure 6.4: Search engine perspective