Verbum XL, 2018, n° 2, 199-220

APPLICATION DES GRAMMAIRES CATÉGORIELLES À L'ANALYSE DE LA QUALITÉ DES EXIGENCES INDUSTRIELLES

Juyeon KANG
Semios for Requirements

Jungyeul PARK
University at Buffalo

RÉSUMÉ
Cet article introduit une méthode pour l'analyse de la qualité des exigences industrielles en utilisant des grammaires catégorielles combinatoires et applicatives GCCA (Grammaire Catégorielle Combinatoire Applicative). Il traite d'exemples formulés en anglais, en français et en coréen pour décrire les quantifications universelles et existencielles. Il propose un système automatique expérimental, CCG-supertagging, avec un traitement syntaxique et la sémantique formelle dans le cadre de la GCCA, des exigences industrielles centrées sur les opérations de quantification en vue de l'extraction des exigences et de la génération des structures sémantiques fonctionnelles.

ABSTRACT
This paper introduces a method for analyzing requirements quality using applicative and combinatory categorial grammars (ACCG). It presents a new way to represent universal and existential quantifiers, with several examples from English, French and Korean, to describe requirements quality. It also proposes the experimental automatic formal syntactic and semantic system, CCG supertagging, based on the ACCG from requirements quality in natural language oriented to the quantification operations in extracting requirements, and a generation of semantic functional structures.

1. INTRODUCTION ET CONTEXTE

Les exigences forment un genre spécifique qui précise ce qu'un système doit faire, avec des actions à effectuer : les interprétations ambiguës doivent être évitées au maximum. Lorsqu'on rédige les exigences, il est

difficile de s'assurer que les textes soient lisibles et sans ambiguïté. Les expériences montrent que, même après plusieurs relectures, la plupart des textes contiennent encore des erreurs terminologiques, grammaticales, discursives et stylistiques par rapport aux bonnes pratiques de rédaction. Selon le rapport publié par *The Standish Group*[1], 50% des projets en difficulté le sont à la suite d'erreurs liées aux exigences, et 70% de difficultés de compréhension viennent des exigences non déclarées et implicites (Lauesen et Vinter 2014). Il est donc important d'identifier les risques potentiels dans les documents techniques dans différents domaines comme l'aéronautique, l'automobile, l'énergie, les constructions navales, les télécommunications... Les risques, liés non seulement à la santé et l'écologie mais aussi aux problèmes sociaux et économiques, sont causés par des spécifications mal rédigées et par différentes formes d'incohérence entre certains éléments. Dans les documents techniques[2] soumis aux exigences, les textes doivent respecter des contraintes de rédaction très strictes, par exemple, non-ambiguïté, complétude, lisibilité, faisabilité, traçabilité, testabilité, cohérence et cohésion. Ces dernières ont des impacts métiers directement observables dans les expressions linguistiques (aux niveau grammatical, lexical ou stylistique) (Figure 1).

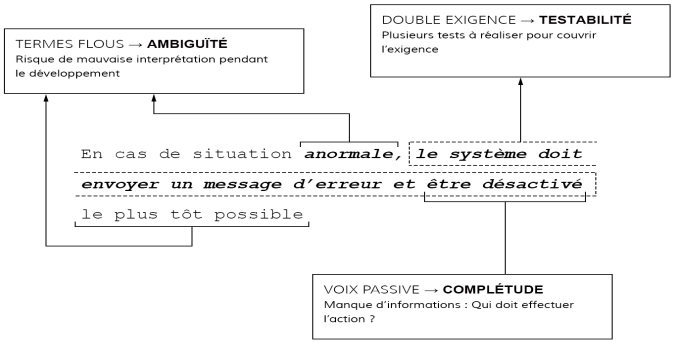


Figure 1. – Impacts métier des erreurs dans une exigence

Nous résumons, dans la Table 1, les problèmes linguistiques les plus fréquemment rencontrés dans les spécifications.

1 The Standish Group, Chaos Manifesto 2018.

2 *Controlled natural language requirements in the design and analysis of safety critical I&C systems*, Research Report, SAREMAN Project, 2013.

Les exigences se présentent souvent sous forme de listes ou bien de phrases composées identifiées et cadrées par un ID (par exemple, [REQ-SYS_0001]). Elles sont structurées en paragraphes. Une exigence se compose d'une ou de plusieurs phrases, qui en définissent par exemple les contraintes ou les différentes facettes (Grady 2013 ; Hull *et al*. 2011). Les règles de la rédaction qui s'appliquent aux exigences sont définies dans les normes IEEE, par des recommandations spécifiques aux exigences et aux principes des langues contrôlées : Normes IEEE830-1998 ; ISO/IEC/IEEE 29148 : 2011 ; ISO/IEC 12207 ; ARP4754 ; ASD-STE 100[3] ; VTT-R-01067-14 (Tommila et Pakonen 2014). Les contraintes de rédaction précisent la syntaxe et la sémantique ainsi que le style et le lexique que les rédacteurs doivent respecter.

Types	**Problèmes identifiés / Fréquence d'erreur**	**Exemples**
Termes flous	Ambiguïté, testabilité / 25%	*wherever possible, suitably, adequately*
Coordination complexe et ou ambiguë	Singularité/ 8%	X *shall* ACTION_1 *and* ACTION_2 *or* ACTION_3
Négation multiple ou marqueurs de négation multiples	Lisibilité/ 8%	*It shall not be possible to do not...*
Quantification universelle ou existentielle	Ambiguïté / 4%	
Multiples actions dans une exigence	Validité, testabilité, traçabilité / 6%	X *shall* ACTION_1 *and* ACTION_2 / X *shall* ACTION_1 *and* Y *shall* ACTION_2
Relatives complexes	Lisibilité / 7%	that x... which... y....
Complexes structures discours	Lisibilité, ambiguïté / 8%	[SUBJECT],-[CONDITION],-[ACTION]-[OBJECT]
Pronoms à référence incertaine	Ambiguïté / 8%	*their, them, these, it...*
Références incorrectes à d'autres chapitres	Faisabilité / 3%	*below, above, see...*
Structure ternaire du verbe d'action	Complétude / 13%	X *shall* $\text{ACTION}_{(send)}$ X *to* Y: *the main system shall send the values of voltages...*
Enumération hétérogène	Charge cognitive élevée, ambiguïté / 10%	1. *system interaction*; 2. *user interface*; 3. *update is not applicable*; → non homogeneous with a and b

Table 1. – Typologie des problèmes linguistiques dans les spécifications

Le présent travail vise à analyser la bonne formation syntaxique et structurelle des exigences en langues naturelles, en faisant appel au modèle catégoriel de la GCCA (Grammaire Catégorielle Combinatoire et Applicative) ; voir Biskri et Desclés (1995, 1996), Kang (2011), Desclés

[3] ASD (Association of European Airlines) – STE100 (Simplified Technical English), Issue 7, 2017.

et al. (2016a,b). Nous nous intéressons également au problème des quantificateurs (quantification universelle ou existentielle portant sur les objets manipulés) qui engendrent des ambiguïtés dans les spécifications. Après avoir fait une étude formelle de certains problèmes de différentes langues, dont l'anglais, le français et le coréen, au moyen du modèle formel, de la GCCA, en y ajoutant certains concepts de la Logique Combinatoire de Curry, nous montrons l'automaticité de ces modèles formels en développant un analyseur morphosyntaxique.

Pour commencer cette étude, nous donnons, au § 2, une brève introduction des Grammaires Catégorielles et de la Logique Combinatoire ; le § 3 sera consacré à la modélisation linguistique des exigences industrielles. Nous y présenterons les problématiques et proposerons une méthode d'analyse formelle susceptible d'y répondre. Dans le § 4, un outil d'analyse de la qualité des exigences sera évoqué et l'architecture générale sera décrite. Nous conclurons, dans le § 5, avec les idées principales abordées dans cette étude, ce qui nous amènera à dégager des perspectives intéressantes pour de futures recherches.

2. GRAMMAIRES CATÉGORIELLES ÉTENDUES ET LOGIQUE COMBINATOIRE

2.1. Introduction au modèle catégoriel

Le choix théorique s'est porté sur le formalisme des Grammaires Catégorielles, présentées par divers auteurs, dont Ajdukiewicz (1935), Curry (1951), Bar-Hillel (1953), Lambek (1961), et fondées sur l'application d'opérateurs à des opérandes de différents types. Différents travaux utilisent les systèmes « opérateur / opérande » introduits à plusieurs reprises par les logiciens, les grammairiens et les linguistes. Parmi les différents modèles catégoriels, nous adoptons le cadre théorique des Grammaires Catégorielles Combinatoires (GCC), qui fait appel à la logique combinatoire de Curry (Curry et Feys 1958), déjà utilisée en linguistique par Shaumyan (1987) et Desclés (1990). Le modèle utilisé est celui de la Grammaire Catégorielle Combinatoire Applicative (GCCA) développé par Biskri et Desclés (2005). La GCCA développe un formalisme « applicatif », dans lequel un opérateur s'applique à un opérande pour construire un résultat qui peut, à son tour, fonctionner comme un opérateur ou comme opérande. Les opérateurs sont catégorisés selon les différents types fonctionnels de Church (1940). En partant de catégories syntaxiques de base et de types fonctionnels assignés aux unités linguistiques (mots ou morphèmes ou syntagmes), le formalisme engendre un véritable « calcul de bonne formation syntaxique ». Ce formalisme est inséré dans des formalismes plus puissants, comme la logique combinatoire typée de Curry (Curry et Feys 1958) ou du λ-calcul de Church (Church 1941), qui trouvent des applications sérieuses dans l'analyse

grammaticale et sémantique des langues naturelles ainsi que dans l'analyse d'une grande famille de langages de programmation. Il possède, par ailleurs, des propriétés mathématiques intéressantes, comme l'isomorphisme de Curry-Howard entre propositions et types fonctionnels.

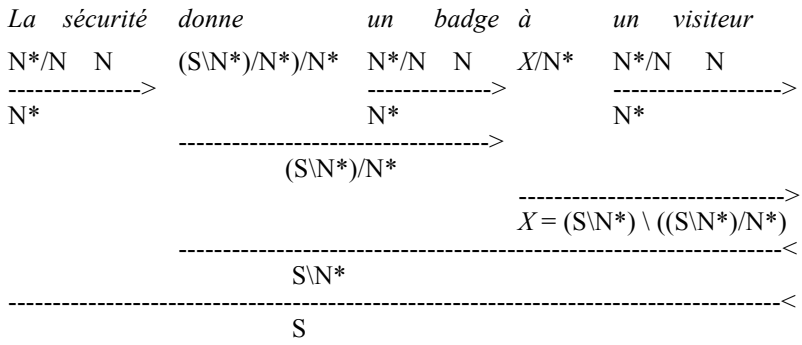
La GCCA (Desclés 1990 ; Biskri and Desclés 1995, 1996, 2005) a été développée depuis 1990 dans le laboratoire LaLIC (Langues, Logiques, Informatique et Cognition) de l'Université Paris-Sorbonne, principalement par les Professeurs Jean- Pierre Desclés et Ismaïl Biskri ; l'un des objectifs de la GCCA est d'une part, l'analyse des phénomènes linguistiques du français comme la coordination et la subordination, et d'autre part, l'automatisation des analyses syntaxiques. Le formalisme de la GCCA est l'un des modules du modèle plus général des Grammaires Applicatives. Ce module assure les premières étapes d'analyse : passer des configurations morpho-syntaxiques de la chaîne syntagmatique, à la construction de représentations logico-grammaticales, afin d'ouvrir une voie pour la construction ultérieure de représentations sémantico-cognitives. Autrement dit, la GCCA est utilisée comme un outil formel qui d'une part, effectue des analyses morpho-syntaxiques et d'autre part, engendre des représentations logico-grammaticales sous-jacentes aux phrases analysées. Avec ces représentations logico-grammaticales des phrases, nous obtenons des représentations qui sont les sources nécessaires pour les analyses sémantico-cognitives formulées dans les modèles de la GAC (Grammaire Applicative et Cognitive) et de la GRACE (GRammaire Applicative Cognitive et Enonciative) ; sur ces modèles, voir : Desclés *et al*. (2011), Desclés et Ro (2011), Desclés *et al*. (2016 b). Les représentations syntaxiques construites par une grammaire catégorielle ont en fait un double objectif : 1°) vérifier la bonne connexion syntaxique d'une expression concaténée qui se présente comme une suite de mots (avec des types assignés) donnée en entrée, et 2°) lorsque cette suite de mots est reconnue comme étant une phrase bien formée, il s'agit de construire la représentation applicative sous-jacente à la phrase donnée en entrée.

Présentons un exemple de vérification de la bonne connexion syntaxique et de la construction de sa représentation applicative, avec la phrase *La sécurité donne un badge à un visiteur*. Une procédure (formulable dans un langage de programmation) établit un changement de représentation qui fait passer automatiquement de la présentation syntagmatique concaténée de la phrase à sa représentation applicative. Ce changement de représentation s'effectue en plusieurs étapes, présentées ci-dessous, par un calcul effectué uniquement en utilisant des règles d'application (à droite et à gauche) qui sont relatives aux types syntaxiques.

Expression concaténée :

La + *sécurité* + *donne* + *un* + *badge* + *à* + *un* + *visiteur*

Pour aider la présentation du calcul, posons [*X* = ((S\N*)\((S\N*)/N*))].



Dans cette analyse catégorielle, à la préposition *à* est associé le rôle syntaxique d'un opérateur qui, en s'appliquant à une expression nominale, construit un opérateur qui peut à son tour s'appliquer à un opérateur verbal transitif, de type (S\N*)/N*, pour construire un prédicat verbal intransitif, de type S\N* ; cet opérateur prépositionnel peut être vu également comme un opérateur relationnel binaire qui établit une relation entre une expression nominale, de type N* et un prédicat transitif, de type (S\N*)/N*, pour construire un prédicat intransitif, de type S\N*, ce qui explique que le type ((S\N*)/(S\N*)/N*))/N* soit assigné à la préposition '*à*'.

L'analyse syntaxique est un processus opératoire qui partant de la représentation syntagmatique concaténée construit une représentation applicative préfixée, décomposable en opérateurs (toujours en position préfixée devant leurs opérande auxquels ils s'appliquent).

2.2. Introduction des combinateurs

Le système des catégories et des types syntaxiques est enrichi par l'introduction d'opérateurs abstraits, appelés combinateurs, qui permettent de composer ou de transformer les types syntaxiques ; l'action opératoire des combinateurs est définie dans le cadre de la Logique Combinatoire de Curry. A titre d'exemple, rappelons les règles d'introduction et d'élimination du combinateur **Φ** :

$$\frac{f(gx)(hx)}{(\Phi fgh)x}\ [\text{intr. }\Phi] \qquad \frac{(\Phi fgh)x}{f(gx)(hx)}\ [\text{elim. }\Phi]$$

Avec un ensemble très réduit de combinateurs, dits « élémentaires », tels que **B** (pour la composition fonctionnelle), **S** (pour la composition par « fusion »), **Φ** (pour l'intrication ou la composition en parallèle), **C*** (pour le changement de type ou « montée du type »), et **C** (pour la permutation des opérandes d'un opérateur binaire)[4], il est possible de traiter différentes organisations syntaxiques complexes au moyen des combinateurs, ce qui conduit à pouvoir formuler les règles de composition et de transformation des types syntaxiques des Grammaires Catégorielles, devenant ainsi des Grammaires Catégorielles Combinatoires (ou étendues). Les combinateurs **B**, **C*, C** et **Φ** , ainsi que les règles applicatives, vont jouer désormais un rôle essentiel dans les grammaires catégorielles étendues. Les combinateurs permettent d'analyser les phrases sans introduire les variables « liées » (à la différence des lambda-expressions) et de traiter des phénomènes langagiers complexes comme la coordination d'éléments de constituants différents, l'ellipse des prédicats, la subordination, la flexibilité de l'ordre des mots, l'agglutination, la thématisation, la dépendance à distance ou croisée, la quantification, etc. (Kang, 2011)[5]. Donnons un exemple d'utilisation des combinateurs en montrant comment une préposition, comme *à*, peut se composer avec un prédicat verbal. Reprenons l'exemple *La sécurité donne un badge à un visiteur*, dont nous avons déjà présenté l'analyse syntaxique et la représentation applicative qui en résulte :

((à (un visiteur)) (donne (un badge))) (la sécurité)

Dans cette expression, la préposition *à* établit une relation entre le terme nominal (*un visiteur*) et le prédicat binaire '(*donne* (*un badge*))' pour construire le prédicat unaire '(*à (un visiteur*)) (*donne* (*un badge*))'. Pour dégager le rôle du troisième actant (*un visiteur*), il faut introduire le combinateur **C** qui permute l'expression adverbiale '(*à* (*un visiteur*))' avec le prédicat verbal '*donne*', afin de pouvoir composer par la suite, à l'aide du combinateur **B**, la préposition '*à*' avec '*donne'*, d'où la définition du le prédicat composé '*donne-à*', obtenu au moyen du combinateur (non élémentaire) '**B**(**CB**)', qui exprime le mode de composition du prédicat verbal '*donne*' avec la préposition '*à*' :

[donne-à = $_{\text{def}}$ **B** (**CB**) donne à].

---

4 Les différentes règles d'introduction et d'élimination (dans le style de la déduction naturelle) des combinateurs sont rappelées dans l'annexe du premier article de ce numéro. Pour une définition plus détaillée des combinateurs élémentaires et complexes, nous renvoyons le lecteur à : Desclés *et al.* 2016a : 30-32 ; 2016b : 194-198, 247 et sq.)

5 Voir des exemples de tels traitements à l'aide des combinateurs dans les articles de J.-P. Desclés et de I. Biskri publiés dans ce même numéro.

La relation entre les deux expressions applicatives est établie par la déduction suivante :

1. *(à (un visiteur)) (donne (un badge)) (la sécurité)*
   ---
2. **B** *(à (un visiteur)) donne (un badge) (la sécurité)*
3. **CB** *donne (à (un visiteur)) (un badge) (la sécurité)*
4. **B** (**CB**) *donne à (un visiteur) (un badge) (la sécurité)*
5. [*donne-à* $=_{\text{def}}$ **B** (**CB**) *donne à*]
   ---
6. *donne-à (un visiteur) (un badge) (la sécurité)*

Cette déduction construit une relation d'équivalence (sémantique) entre deux expressions applicatives :

(à (un visiteur)) (donne (un badge)) (la sécurité)
$\Rightarrow$ *donne-à (un visiteur) (un badge) (la sécurité)*

À la seconde représentation, on peut associer directement un arbre de dépendance, au sens de Tesnière, formé par le prédicat '*donner-à*', positionné au sommet de l'arbre avec ses trois arguments (ou actants) aux feuilles de l'arbre, ces arguments fonctionnent respectivement comme « un receveur » (troisième actant), « un objet échangé » (second actant) et « un donneur » (premier actant).

Depuis Frege, les quantificateurs sont, dans le cas des quantificateurs simples, des opérateurs qui s'appliquent à des prédicats (unaires) pour construire des propositions , par exemple :

Tous (être-mortel) $\cong$ *Tous sont mortels*

Dans le cas de la quantification restreinte, les quantificateurs s'appliquent à des propriétés ou prédicats, associés à des unités nominales (des noms communs) pour construire des opérateurs qui s'appliquent directement à des prédicats verbaux et construire ainsi des propositions, par exemple :

(Tout (être-homme)) (être-mortel) $\cong$ *Tout homme est mortel*

Les quantificateurs restreints, notés Π_2 et Σ_2 sont définis comme étant fonction des quantificateurs simples Π_1 et Σ_1 ; plus précisément :

Quantificateur universel Π_2 :	Quantificateur existentiel Σ_2
$[\Pi_2 = \mathbf{B}(\mathbf{CB}^2)\mathbf{\Phi} \Rightarrow \Pi_1]$	$[\Sigma_2 = \mathbf{B}(\mathbf{CB}^2)\mathbf{\Phi} \wedge \Sigma_1]$

Dans ces équivalences définitoires[6], le combinateur $\mathbf{B}^2$ est la composition de **B** avec **B**, c'est-à-dire [$\mathbf{B}^2 = \mathbf{BBB}$]. Nous allons revenir plus

6 Voir : Desclés *et al.,* (2016 b : 250-267).

loin sur les quantificateurs restreints et leur rôle dans l'analyse des exigences.

3. MODÉLISATION LINGUISTIQUE DES EXIGENCES INDUSTRIELLES

3.1. Problématique

Parmi les problèmes exposés dans le Tableau 1, nous nous concentrons dans cette étude sur la vérification de forme syntaxique des exigences. Sachant que les exigences sont rédigées dans un langage contrôlé, structurellement et syntaxiquement, nous observons quelques problèmes récurrents :

1. oubli d'un argument lors de l'utilisation des verbes ternaires
   a. *The system shall send* (prédicat à deux arguments) *the data* (1er argument) [**A QUI/QUOI**].
   b. *The system shall be tested* [**PAR QUI/QUOI**].
2. utilisation des quantificateurs universels et existentiels
   c. *All system shall provide a visual warning.*
   d. *Some sub-systems shall be activated by the defined software in the document SYS_825.*

Comme le montrent les exemples ci-dessus, les exigences contiennent les éléments qui peuvent provoquer la mauvaise compréhension, et par la suite, réduire la lisibilité et augmenter la charge cognitive des utilisateurs. Dans un cycle de vie d'un produit, cela peut se produire au moment de la conception, du développement et/ou de la vérification / validation. Les expériences auprès des industriels nous confirment le besoin d'identifier les problèmes de façon automatique, sachant qu'il est quasiment impossible de vérifier, exigence par exigence, la qualité de ces textes composés de milliers d'exigences textuelles, et par conséquent, de réduire le coût et le temps d'un développement de systèmes.

Dans le cas d'un oubli d'arguments, nous trouvons deux types de problèmes. L'exigence (a) spécifie l'action de système "envoyer l'information", car la question sur le récepteur de l'information se pose. Le système peut envoyer l'information à une personne ou à un autre système, soit interne, soit externe. Lorsqu'on veut traiter cette exigence, selon le concepteur, elle peut être interprétée différemment, ce qui est source d'ambiguïté. L'exigence (b) est considérée comme incomplète car il manque l'information importante sur l'agent, l'acteur de cette exigence. Par conséquent, les concepteurs doivent faire appel à des connaissances extérieures pour la comprendre. Cela augmente le temps de conception et la charge cognitive des ingénieurs.

Un autre phénomène qui est naturellement analysé en termes des relations de dominance au niveau de la forme logique est celui de la portée

des quantificateurs (*quantifier scope* en anglais). Il est normal d'assumer que l'ambiguïté de la phrase *Every boy admires some saxophonist*[7] doit être représentée sous deux formes logiques qui se distinguent par la portée des quantificateurs.

Every boy admires some saxophonist.

a. $\forall x.\text{boy}'x \rightarrow \exists y.\text{saxophonist}'y \wedge \text{admires}'yx$
b. $\exists y.\text{saxophonist}'y \wedge \forall x.\ \text{boy}'x \rightarrow \text{admires}'yx$

La question se pose de savoir comment la grammaire peut assigner les interprétations correctes aux phrases contenant des quantifications multiples. Avec les notations et les hypothèses de la GCC (Steedman, 2000), la façon proposée pour intégrer les quantificateurs généralisés dans la sémantique des déterminants des Grammaires Catégorielles consiste à transférer le changement de type au lexique. On assigne les types définis ci-dessous aux déterminants comme *every* et *some*. Cela revient à transformer ces quantificateurs typés en opérateurs qui s'appliquent chacun à un terme nominal. Les quantificateurs généralisés typés se présentent comme des opérateurs typés (où T désigne le type des propositions, N le type des noms et NP le type des syntagmes nominaux) formulés dans le lambda-calcul :

1. every = (T/(T\NP))/N : $\lambda p.\ \lambda q.\ \forall x.(px \rightarrow qx)$
2. every = (T\(T/NP))/N : $\lambda p.\ \lambda q.\ \forall x.(px \rightarrow qx)$
3. some = (T/(T\NP))/N : $\lambda p.\ \lambda q.\ \exists x.(px \wedge qx)$
4. every = (T\(T/NP))/N : $\lambda p.\ \lambda q.\ \exists x.(px \wedge qx)$

Étant donné les types syntaxiques présentés ci-dessus, les dérivations que la GCC permet assignent deux formes logiques distinctes correspondant à *every* et à *some*. Dans le cas de l'utilisation des quantificateurs, la portée de ces quantificateurs est également souvent ambiguë dans les exigences. C'est la raison pour laquelle les standards et normes de la rédaction des exigences déconseillent la présence des quantificateurs, soit universels, soit existentiels.

3.2. Analyse de quelques exemples

Nous proposons dans cette partie les analyses formelles des exigences industrielles venant des spécifications rédigées en plusieurs langues : anglais, français et coréen. Notre approche permet dans un premier temps d'analyser et de représenter les phénomènes linguistiques précis et variés de deux familles de langue très distinctes, en minimisant le problème des ambiguïtés possibles, et dans un deuxième temps, en

7 L'exemple est extrait de Steedman (2000 : 71).

dégageant certains invariants relatifs aux opérations de prédication, de détermination, de transposition, de quantification, etc.

3.2.1. Exemple d'analyse de l'exigence en anglais

Nous donnons l'analyse formelle d'une exigence standard contenant un sujet *the APU* suivi d'un modal *shall*, d'un verbe d'action *provide* et d'un objet *the visual warning*, c'est-à-dire : *The APU shall provide the visual warning.* Cette analyse se fait en deux temps : 1°) l'étape des calculs catégoriels et 2°) l'étape d'élimination des combinateurs, ce qui permet de construire la représentation applicative de la phrase. Dans l'étape des calculs catégoriels opérant sur les types, des types syntaxiques ont été assignés aux unités linguistiques. Sachant que les types syntaxiques de base sont S pour la phrase, N pour les nominaux et N* pour les syntagmes nominaux complets, nous assignons au nom *APU* le type N, et au déterminant *the* le type (N*/N). Le type (N*/N) signifie que le déterminant *the* attend à sa droite une expression de type N (*APU*) pour construire le syntagme nominal complet de type N* (*the* (*APU)*). Nous appliquons la règle de changement de type (associée au combinateur **C***) à (*the* (*APU*)), pour donner à cette expression le rôle d'un opérateur, de type S/(S\N*), qui peut alors se composer, en introduisant le combinateur **B,** avec le verbe (*shall*)*provide*, de type (S\N*)/N*. Dans l'étape d'élimination des combinateurs, nous éliminons d'abord le combinateur **B** qui composait les deux expressions typées (*the APU*) et (*shall*) *provide*. L'élimination de **C*** permet ensuite de ramener l'expression (*the APU*) au rôle syntaxique d'un opérande de l'opérateur ((*shall*) *provide* (*the* (*visual warning*))) pour construire la représentation applicative de cette phrase structurée uniquement par des applications d'opérateurs à des opérandes.

The *APU* *(shall) provide* *the* *visual* *warning*

(N*/N) N (S\N*)/N* (N*/N) (N/N) N

----------> ----------->

N* N : (visual warning)

---------- **C*** ----------->

S/(S\N*) : **C***(the(APU)) N* : (the(visual warning))

----------**B**

(S/N*) : **B**(**C***(the(APU))((shall)provide))

---------->

S : **B**(**C***(the(APU))((shall)provide)) (the(visual warning))

---------- [elim. **B**]

S : **C***(the(APU))(((shall)provide) (the(visual warning)))

---------- [elim. **C***]

S : ((((shall)provide) (the(visual warning))) (the(APU)))

Reprenons l'exemple (1a) présenté au § 3.1, afin de montrer la détection du problème de complétude : *The system shall send the data [A QUI/QUOI]*

The *system* *(shall)send* *the* *data*
(N*/N) N ((S\N*)/N*)/N* (N*/N) N
---------->
N* : (the(system))
---------- C* ---------->
(S/(S\N*)) :**C***(the(system)) N*: (the(data))
---------->
(S\N*)/N* : (((shall)send)(the(data)))
---------- **B**
(S/N*) : **B**(**C***(the(system))) (((shall)send) (the(data)))

Le calcul est bloqué : échec de l'analyse.

Le verbe d'action *send* de cette exigence nécessite trois arguments pour réaliser proprement l'action spécifiée. Le calcul catégoriel précédent amène l'analyse à l'expression typée de (S/N*), ce qui signifie que, dans cette exigence, il manque un argument nominal, de type N*, pour former une phrase syntaxiquement correcte : l'analyse est bloquée.

Passons maintenant à l'analyse de quantificateurs, avec l'exemple (2c) : *All systems shall provide a visual warning* (voir p. suiv.).

Dans cette analyse, nous analysons *all* comme un quantificateur universel restreint Π_2, de type (S/(S\N*)/N, et *a* comme un quantificateur existentiel restreint Σ_2, de type ((S\N*)\((S\N*)/N*))/N. Du point de vue sémantique, cette quantification peut être interprétée de la façon suivante : pour tout *y* qui tombe sous le prédicat *system*, il existe un *x* qui tombe sous le prédicat *visual warning*.

Les structures syntaxiques des langues peuvent-elles être décrites avec les mêmes concepts formels et les mêmes modes de composition alors que les structures syntaxiques sont entièrement spécifiques à chaque langue ? La comparaison, sur les exemples, de l'anglais, du coréen et du français, est un excellent champ d'expérimentation qui devrait permettre de mettre à l'épreuve notre approche sur le plan, à la fois, syntaxique et sémantique.

All *systems* *(shall)provide* *a* *visual* *warning*

(S/(S\N*))/N N (S\N*)/N* ((S\N*)\((S\N*)/N*))/N N/N N

----------> ---------->

(S/(S\N*)) : all (systems) N:(visual(warning))

----------> ((S\N*)\((S\N*)/N*)) : a (visual (warning))

<---------- (S\N*) : (a (visual (warning))) ((shall)provide)

----------> S : all (systems) ((a (visual (warning))) ((shall)provide))

---------- [all = Π_2]

Π_2 (systems) ((a (visual(warning))) ((shall)provide))

---------- [$\Pi_2 = \mathbf{B}(\mathbf{CB}^2)\mathbf{\Phi} \Rightarrow \Pi_1$]

$\mathbf{B}(\mathbf{CB}^2)\mathbf{\Phi} \Rightarrow \Pi_1$ (systems)((a (visual(warning))) ((shall)provide))

---------- [elim. **B**]

$(\mathbf{CB}^2)(\mathbf{\Phi} \Rightarrow \Pi_1$ (systems)((a (visual(warning))) ((shall)provide))

---------- [elim. **C**]

$(\mathbf{B}^2)\ \Pi_1\ (\mathbf{\Phi} \Rightarrow)$ (systems) ((a (visual(warning))) ((shall)provide))

---------- [elim. $\mathbf{B}^2$]

Π_1 $((\mathbf{\Phi} \Rightarrow)$ (systems) ((a (visual(warning))) ((shall)provide)))

---------- [elim. Π_1]

$(\mathbf{\Phi} \Rightarrow$ (systems)((a (visual(warning))) ((shall)provide))) y

---------- [elim. **Φ**]

$\Rightarrow$ ((systems)(y)) ((a(visual(warning))) ((shall)provide))(y))

---------- [a = Σ_2]

$\Rightarrow$ ((systems)(y)) ((Σ_2 (visual(warning))) ((shall)provide))(y))

---------- [$\Sigma_2 = \mathbf{B}(\mathbf{CB}^2)\mathbf{\Phi} \wedge \Sigma_1$]

$\Rightarrow$ ((systems)(y))((**B**(**CB**^{2}) **Φ**$\wedge\Sigma_1$ (visual(warning))) ((shall)provide))(y))

---------- [elim. **B**]

$\Rightarrow$ ((systems)(y))(((**CB**^{2})(**Φ** $\wedge$) Σ_1(visual(warning))) ((shall)provide))(y))

---------- [elim. **C**]

$\Rightarrow$ ((systems)(y)) (((**B**^{2})Σ_1(**Φ** $\wedge$)(visual(warning))) ((shall)provide))(y))

---------- [elim. $\mathbf{B}^2$]

$\Rightarrow$ ((systems)(y)) ((Σ_1((**Φ** $\wedge$)(visual(warning))) ((shall)provide))(y)))

---------- [elim. Σ_1]

$\Rightarrow$ ((systems)(y)) (((**Φ** $\wedge$ (visual(warning))) ((shall)provide))(y)))x

---------- [elim. **Φ**]

$\Rightarrow$ ((systems)(y)) ($\wedge$ ((visual(warning))(x))(((shall)provide) (y)(x)))

3.2.2. Exemple d'analyse de l'exigence en français

Nous analysons le rôle du quantificateur *tous* dans le cadre de la GCCA. L'exigence ci-dessous a été extraite d'une spécification industrielle écrite en français : *Tous les capteurs doivent être endurcis.*

Tous les capteurs doivent être endurcis
(S/(S\N*))/N* N*/N N ((S\N*)/(N\N))/((S\N*)/(N\N)) (S\N*)/ (N\N) (N\N)
---------->
N*: (les capteurs)
---------->
(S/(S\N*)) : (tous(les capteurs))
---------->
((S\N*)/(N\N)) : (doivent être)
---------->
((S\N*) : ((doivent être) endurcis)
---------->
S : (tous(les capteurs)) ((doivent être) endurcis)
---------- [tous = Π_2]
Π_2 (les capteurs) (doivent (être endurcis))
---------- [$\Pi_2 =_{def}$ **B**(**CB**^{2})**Φ** => Π_1]

B(**CB**^{2})Φ => Π_1 (les capteurs) ((doivent être) endurcis)
---------- [elim. **B**]
(**CB**^{2}) (**Φ** =>) Π_1 (les capteurs) ((doivent être) endurcis)
---------- [elim. **C**]
B^2 Π_1 (Φ =>) (les capteurs) ((doivent être) endurcis)
---------- [elim **B**^{2}]
Π_1 ((**Φ** =>) (les capteurs) ((doivent être) endurcis))
---------- [elim. Π_1]
(**Φ** => (les capteurs) ((doivent être) endurcis)) x
---------- [elim. **Φ**]
=> ((les capteurs) (x)) ((doivent être) endurcis) (x)
---------- [notations infixées]
[((les capteurs) (x)) => ((doivent être) endurcis)(x)]

((les capteurs) (a)) [hypothèse]
---------- [*modus ponens*]
((doivent être) endurcis)(a)

En français, le quantificateur *tous* apparaît souvent dans les syntagmes nominaux quantifiés comme dans *tous les capteurs*. L'expression *les capteurs* construit un syntagme nominal complet de type N* et le quantificateur *tous* s'applique au syntagme nominal (*les capteurs*). L'analyse dans la GCCA traite les adjectifs comme des déterminants de termes nominaux et le quantificateur *tous* comme un opérateur de type (S/(S\N*))/N*. Dans le cadre de la GCCA, le quantificateur *tous* du français peut être directement représenté sous la forme d'un déterminant nominal, ce qui conduit alors à une analyse non-frégéenne des syntagmes nominaux quantifiés. Sur la distinction entre les analyses par la

« quantification frégéenne » et par la « quantification non-frégéenne », nous renvoyons le lecteur à Desclés (2005) et à Desclés *et al.* (2016b).

3.2.3. Exemple d'analyse de l'exigence en coréen

La langue coréenne se caractérise comme étant agglutinante avec des morphèmes très nombreux. En coréen, les suffixes qui apparaissent à la fin des noms, et les suffixes verbaux, qui apparaissent après les racines verbales, déterminent les fonctions grammaticales de chaque élément linguistique d'une phrase. Dans la grammaire coréenne, les suffixes nominaux peuvent engendrer différentes utilisations du même verbe ou d'un nom. La prise en compte des suffixes nominaux est devenue nécessaire dans le cadre de la GCCA pour une analyse adéquate du coréen.

À titre d'exemple, prenons, la phrase suivante du coréen :

abeoji-neu	*Séoul-eso*	*saeop-eu*	*ha-ss-eoss-da*
Père-topique	Séoul-lieu	travail-acc	faire-honorifique-passé-déclaratif

'Mon père faisait son travail à Séoul'

Prenons maintenant un exemple d'exigence en coréen contenant les suffixes nominaux : *-eun*, *-e* et *-eul* analysés dans le cadre de la GCCA. Le coréen, comme d'autres langues à cas non indo-européennes, a des structures très différentes de celles des langues indo-européennes. Bien que l'ordre des éléments soit variable, une exigence est exprimée sous un format standard : SUJET, MODAL, ACTION, OBJET, CONTRAINTE. Prenons l'exemple suivant :

system-eun	*12-gaewo inae-e*	*guchuk-eul*	*wanryoha-eyoya*	*ha-n+da*
système-Top	12-mois dans-Temps	développement-Acc	finaliser-Mod	Faire-Pre+Dcl

'Le système doit être développé dans les 12 mois'

Dans l'analyse qui va suivre, le cas adverbial *-e*, qui accompagne le temps, se voit attribuer le type syntaxique (S\N*)/(S\N*)\N. Ce type montre que le marqueur *-e* attend un nominal de type N pour construire un déterminant du verbe de type (S\N*)/(S\N*) qui fonctionne alors comme un adverbe (*-e* (*inae* (12 *gaewol*))). La première étape d'analyse du calcul catégoriel permet de vérifier la bonne connexion syntaxique de la phrase, et d'autre part, d'analyser les suffixes de cas du coréen analysés comme des opérateurs avec des types syntaxiques fonctionnels. La deuxième étape élimine les combinateurs introduits au cours de l'analyse syntaxique, cette étape d'élimination nous permet de passer à une représentation logico-grammaticale de la configuration morpho-syntaxique initiale.

Analyse catégorielle de la phrase coréenne :

System-eum -12 -gaewol -inae -e -guchuk -eul -wanryohaeyoya -handa

N (N*\N) (N/N)/(N/N) (N/N) N ((S\N*)/(S\N*))\N N (N*\N) (S\N*)\N ((S\N*)\N*)\((S\N*)\N*)
----------> ---------->
N*:(-eun system) N/N
---------->
N: (-inae(12 gaewol))
---<
((S\N*)/(S\N*)) : (-e(-inae(12 gaewol)))

---------------------<
N*: (-eul guchuk)
---<
(S\N*)\N*: (-handa wanryohaeyoya)
------------------------------------- **C*** -- **C***
(S/(S\N*)) : (**C***(-eun system)) (S\N*)/((S\N*)\N*):(**C***(-eul guchuk))
-->
(S\N*): (**C***(-eul guchuk))(-handa wanryohaeyoya)
-->
(S\N*) : ((-e(-inae(12 gaewol))) (**C***(-eul guchuk)) (-handa wanryohaeyoya))
-->
S: (**C***(-eun system)) ((-e(-inae(12 gaewol))) (**C***(-eul guchuk)) (-handa wanryohaeyoya))
-- [elim. **C***]
S: ((-e(-inae(12 gaewol))) (**C***(-eul guchuk)) (-handa wanryohaeyoya)) (-eun system)
-- [elim. **C***]
S: ((-e(-inae(12 gaewol))) (-handa wanryohaeyoya) (-eul guchuk)) (-eun system)

4. PROPOSITION D'UN SYSTÈME D'ANALYSES SYNTAXIQUES ET SÉMANTIQUES FORMELLES

L'implémentation de la GCCA dans le développement de l'analyseur morpho-syntaxique, appelé *Categorial parser*, permet de prendre en compte de façon automatique, dans les analyses qu'il engendre, des phénomènes langagiers que l'on trouve dans les spécifications techniques, tels que les problèmes liés à la complétude et à la quantification. Il engendre également les interprétations sémantiques sous forme d'expressions applicatives, c'est-à-dire de « formes normales » (sans aucunes occurrences de combinateurs) dans le sens de Church-Rosser. Il permet également de réduire certains problèmes d'ambiguïtés (par exemple l'ambiguïté fallacieuse et l'ambiguïté lexicale et structurale) en adoptant la stratégie de « calcul quasi-incrémental de gauche à droite ».

4.1. Workflow (l'architecture opérationnelle) du système

Nous décrivons en six étapes la procédure de traitement général de notre système :

1. Entrée : Documents spécifications. Tout type de documents de spécification peut être considéré.

INPUT : spécifications

To add all housekeeping storage selection definitions that can be generated by an application process, N2 shall be set to 0. TC[15,30] delete some housekeeping storage selection definetions for a packet store. Each telecommand packet transporting a request to delete some housekeeping storage selection definitions for a packet store shall be of message subtype 30. For each telecommand packet transporting a request to delete some housekeeping storage selection definitions for a packet store, the application data field shall have the structure specified in …

2. Pré-traitement : Extraction des exigences. Dans un document de spécification, nous trouvons beaucoup d'autres éléments que des exigences, par exemple : tables de matières, glossaires, notes, introduction, conclusion, etc. Sachant que les phrases de ces parties citées ci-dessus n'ont pas la même importance que les exigences, nous n'extrayons que les exigences pour les garder dans un nouveau fichier d'entrée. Cette extraction est basée sur les patterns prototypiques des exigences.

OUTPUT : liste des exigences extraites

requirements-list.txt	
RE1	The APU shall provide a mechanical power
RE2	When detail reporting time-based scheduled activities, the time-based scheduling subservice shall generate a single time-based schedule detail report.
RE3	The notifications contained in a time-based schedule detail report shall be ordered according to the release time of the reported scheduled activities.
RE4	All systems shall provide two types of visual warning.

3. Typage catégoriel : CCG Bank et Depccg. Il s'agit de tester quelques outils d'attribution des types, nommés « supertags », à chaque unité linguistique des textes. Entre autres, nous utilisons depccg (Yoshikawa *et al.* 2017) et ccg-supertagging (Garrette *et al*. 2014).

INPUT : The APU shall provide a mechanical power.
OUTPUT : Supertags généré par Depccg

```
1 The NP[nb]/N
2 APU N
3 shall (S[dcl]\NP)/(S[b]\NP)
4 provide (S[b]\NP)/NP
5 the NP[nb]/N
6 mechanical N/N
7 power N
```

Conversion de supertags en format parenthésé

```
1 The Sur(NP,N)
2 APU N
3 shall Sur(Sous(S,NP),Sous(S,NP))
4 provide Sur(Sous(S,NP),NP)
5 the Sur(NP,N)
6 mechanical Sur(N,N)
7 power N
```

4. Calcul catégoriel combinatoire

(1) Algorithme présenté dans la thèse de Kang (2011) :

OUTPUT : Exigence analysée sous forme d'arbre et d'expression combinatoire

Calcul catégoriel	
Type syntaxique :	S
Arbre applicatif :	Node (Node (C*, Node (Feuille=the, Feuille=APU)), Node (Feuille=shall, Node (Feuille=provide, Node (Feuille=the, Node (Feuille=mechanical, Feuille=power)))))
Expression combinatoire :	((C* (the APU)) (shall (provide (the (mechanical power)))))

OUTPUT FINAL : Exigence sous forme de structure sémantique fonctionnelle

Structure applicative :	(shall, (provide, (the, (mechanical, (power)))), (the, (APU)))

(2) Calcul catégoriel engendré par depccg

```
The      APU       shall            provide      the   mechanical  power
NP[nb]/N N (S[dcl]\NP)/(S[b]\NP)/NP (S[b]\NP)/NP NP[nb]/N  N/N     N
---------->
 NP ->                                                    ------------->
                                                             N ->
                                                 ------------------------>
                                                         NP ->
                                       ------------------------------------>
                                            S[b]\NP ->
                     ------------------------------------------------------>
                         S[dcl]\NP ->
------------------------------------------------------------------------<
                     S[dcl] ->
```

5. Calcul des opérateurs de quantification : Définition des quantificateurs à l'aide de combinateurs.

INPUT : **All** systems (shall) provide **a** visual warning
OUTPUT :

⇒ ((systems)(y)) (∧ ((visual(warning))(x))(((shall)provide) (y)(x)))

Afin de pouvoir appliquer les règles de réécriture permettant d'engendrer une interprétation logique d'une exigence, nous avons implémenté les patterns en expressions régulières. Ils remplacent les opérateurs Π_2 et Σ_2 par leurs définitions respectives, $[\Pi_2 =_{def} \mathbf{B}(\mathbf{CB}^2)\mathbf{\Phi} => \Pi_1]$ ainsi que $[\Sigma_2 =_{def} \mathbf{B}(\mathbf{CB}^2)\mathbf{\Phi} \wedge \Sigma_1]$.

Cette étude est une expérimentation de l'application du formalisme des Grammaires Catégorielles et de la Logique Combinatoire ; elle nous montre les premiers résultats de l'automatisation de l'analyse formelle des exigences industrielles. Nous avons pu constater que notre approche est applicable à des spécifications écrites en langues structurellement distinctes comme l'anglais, le français et le coréen ; les résultats d'analyse semblent très prometteurs.

5. CONCLUSION ET PERSPECTIVES

L'étude que nous avons menée dans ce travail a pour objectif d'appliquer à la qualité des exigences industrielles une approche d'analyse basée sur les Grammaires Catégorielles et la Logique Combinatoire. Pour cela, nous avons signalé l'importance de la qualité textuelle dans les documents techniques, notamment pour les spécifications industrielles et nous avons décrit les problèmes linguistiques présents dans ces documents. Afin d'y répondre, nous avons proposé une modélisation des exigences dans le cadre de la GCCA. Ce dernier nous a permis de vérifier une bonne forme syntaxique des exigences, notamment en permettant d'aborder les problèmes de complétude et de quantification. Comme le montrent les exemples présentés dans le § 3, cette approche est applicable non seulement à une langue indo-européenne (comme l'anglais et le français) mais aussi à une famille de langue non indo-européenne (comme le coréen). Il serait également intéressant de l'adapter à des documents de spécification écrits en d'autres langues asiatiques (comme le japonais en sachant que le japonais a une structure syntaxique assez similaire à celle du coréen). Cette application a été déjà confirmée dans plusieurs études publiées (Saito 1992 ; Kubota 2008 ; Kang 2011 ; Bekki 2015), même si l'application était réalisée pour des textes généraux.

Pour répondre aux besoins industriels de l'outillage, nous avons mis en place le système *Categorial Parser* d'analyses catégorielles qui donne une première estimation sur la qualité des exigences aux ingénieurs concernés par ce type de documents techniques. Les exigences calculées de type **S** pourront être considérées comme étant syntaxiquement bien formées. Au niveau de l'outillage, en raison d'une indisponibilité des ressources, notamment *CCG Bank* pour toutes les langues, et sachant que *CCG Bank* joue un rôle important dans l'attribution des types catégoriels permettant de passer aux calculs catégoriels, les documents en anglais ont été traités jusqu'au traitement informatique. Nous disposons des autres

modules, excepté *CCG supertagger*, à savoir : l'extraction des exigences, l'algorithme des calculs catégoriels combinatoires et le programme de réécriture adapté au français et au coréen.

En ce qui concerne l'analyse des exigences, nous avons préalablement mené quelques études sur la méthode d'anonymisation (Kang et Park 2018) et la méthode d'analyse de la qualité des exigences par apprentissage automatique (Kang et Park 2016). En effet, les documents techniques contiennent souvent les informations sensibles susceptibles de dévoiler des secrets industriels. Pour utiliser ce type de données confidentielles dans un projet du TAL (Traitement Automatique des Langues), l'anonymisation offre une solution permettant d'exploiter ces données anonymisées sans faire perdre ses qualités. Le travail qui vient d'être présenté utilise les données anonymisées obtenues la méthode proposée dans (Kang et Park, 2018). Nous pensons que ce travail complète les résultats proposés dans (Kang et Park 2016), où nous avons travaillé sur les problèmes d'ambiguïté, de lisibilité, de singularité, de complétude et de conformité , voir le Tableau 1 dans Kang et Park (2016). Les résultats permettent de détecter les problèmes lexicaux et syntaxiques, mais les quantificateurs ou les prédicats à n-argument n'étaient pas pris en compte. Le traitement par la GCCA présenté dans le présent article vient compléter les lacunes de l'approche par apprentissage automatique.

REMERCIEMENTS

Cette étude a bénéficié d'un soutien financier de la région Occitanie dans le cadre d'un projet CLE (Contrat Laboratoire – Entreprise), intitulé ELENAA (des Exigences en LanguEs Naturelles à leurs Analyses Automatiques).

RÉFÉRENCES

AJDUKIEWICZ K. (1935). Die syntaktische Konnexität. *Studia Philosophica* 1, 1-27. Trad. anglaise: Syntactic connexion. In: S. McCall (ed.), *Polish Logic, 1920-1939.* Oxford: Oxford University Press, 1967, 207-231.

BAR-HILLEL Y. (1953). A quasi-arithmetical notation for syntactic description. *Language* 29, 47-58.

Bekki D. (2015). Two types of Japanese scrambling in combinatory categorial grammar. In: *Proceedings for ESSLLI 2015 Workshop 'Empirical Advances in Categorial Grammar' (CG 2015)* 1-12, Barcelona, Spain.

BISKRI I., DESCLÉS J.-P. (1995). Applicative and Combinatory Categorial Grammar (from syntax to functional semantics). *Recent Advances in Natural Language Processing*. Amsterdam: John Benjamins, 71-84.

BISKRI I., DESCLÉS J.-P. (1996). La Grammaire Catégorielle Combinatoire Applicative. *Traitement Automatique des Langues* 37(2), 71-84.

BISKRI I., DESCLÉS J.-P. (2006). Coordination de catégories différentes en français. *Faits de langues* 28, 57-66.

CHURCH A. (1940). A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic* 5(2), 56-68.

CHURCH A. (1941). *The Calculi of Lambda-Conversion*. Princeton University Press.

CURRY H.B. (1951). *Outlines of a Formalist Philosophy of Mathematics*. Amsterdam: North-Holland.

CURRY H.B., FEYS R. (1958). *Combinatory Logic*, vol. I. Amsterdam: North Holland.

DESCLÉS J.-P. (1990). *Langages applicatifs, langues naturelles et cognition*. Paris : Hermès.

DESCLÉS J.-P. (2005). Une analyse non frégéenne de la quantification. In : P. Joray (éd.), *La quantification dans la logique moderne*. Paris : L'Harmattan, 263-312.

DESCLÉS J.-P., RO H.-J. (2011). Opérateurs aspecto-temporels et logique combinatoire. *Mathématiques et Sciences Humaines*, 39-70.

DESCLÉS J.-P., RO H.-J. et KANG J. (2011). De la syntaxe à l'analyse aspecto-temporelle automatique d'un texte par la Grammaire Catégorielle et la Logique combinatoire. In : *Les Actes de 79e Congrès de l'Association Francophone pour le Savoir (ACFAS),* Sherbrooke, Canada.

DESCLÉS J.-P., GUIBERT G. et SAUZAY B. (2016a). *Logique combinatoire* et λ-calcul : des logiques d'opérateurs. Toulouse : Cépaduès.

DESCLÉS J.-P., GUIBERT G. et SAUZAY B. (2016b). *Calcul des* significations par une logique d'opérateurs. Toulouse : Cépaduès.

GARRETTE D., DYER C., BALDRIDGE J. et SMITH N.A. (2014). Weakly-Supervised Bayesian Learning of a CCG Supertagger. In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, 141-150, Ann Arbor, Michigan.

GRADY J.O. (2013). *System Requirements Analysis*. 2° edition, Elsevier.

HULL E., JACKSON K. et DICK J. (2011). *Requirements Engineering*. London: Springer-Verlag.

KANG J. (2011). *Problèmes morpho-syntaxiques analysés dans un modèle catégoriel étendu : application au coréen et au français avec une réalisation informatique.* Thèse de Doctorat. Université de Paris-Sorbonne, France.

KANG J., PARK J. (2016). Generating a Linguistic Model for Requirement Quality Analysis. In: *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Posters (PACLIC 30)*, 439-447, Seoul, Korea.

KANG J., PARK J. (2018). Data Anonymization for Requirements Quality Analysis: a Reproducible Automatic Error Detection Task. In: *Proceedings*

of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan.

KUBOTA Y. (2008). Solving the Morpho-Syntactic Puzzle of the Japanese -Te Form Complex Predicate: A Multi-Modal Combinatory Categorial Grammar Analysis. In: *Proceedings of The Seventh Syntax and Semantics Conference in Paris (Empirical Issues in Syntax and Semantics 7)*. Paris, France, 283-306.

LAMBEK J. (1961). On the calculus of syntactic types. In: R. Jakobson (ed.), *Structure of language and its mathematical aspects*. *Proceedings of symposia in applied mathematics*. Providence, Rhode Island: American Mathematical Society, 166-178.

LAUESEN S., VINTER O. (2014). Preventing Requirement Defects: An Experiment in Process Improvement. *Requirements Engineering* 6(1), 37-50.

SAITO M. (1992). Long distance scrambling in Japanese. *Journal of East Asian Linguistics* 1(1), 69-118.

SHAUMYAN S. (1987). *A Semiotic Theory of Language*. Bloomington: Indiana University Press.

STEEDMAN M. (2000). *The Syntactic Process*. The MIT Press.

TOMMILA T., PAKONEN A. (2014). Controlled natural language requirements in the design and analysis of safety critical I&C systems. Technical report, VTT.

YOSHIKAWA M., NOJI H. et MATSUMOTO Y. (2017). A* CCG Parsing with a Supertag and Dependency Factored Model. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers). Vancouver, Canada, 277-287.