Transduction sémantique pour la modélisation de système

A. Lamercerie
Univ Rennes, Inria, Irisa

22 juin 2020

Résumé

La conception d'un système est un défi pour les ingénieurs. Le coût d'une erreur s'avère parfois considérable. De fait, leurs détections est un enjeu important. Des méthodes de vérification formelles ont été élaborées pour y répondre. Cependant, l'écart est important entre la promesse théorique de ces méthodes et leur intégration pratique, un écart que cet article vise à réduire. La formalisation automatique d'énoncés exprimés en langue naturelle est explorée par la proposition d'une méthode associant des approches statistiques et symboliques.

Mots-clés

Conception de système, analyse sémantique, représentation abstraite, formalisation.

Abstract

The system design is a challenge for engineers. The cost of an error is sometimes considerable, and their detection is an important issue. Formal verification methods have been developed to address this. However, there is a significant gap between the theoretical promise of these methods and their practical integration, a gap that this article intends to reduce. The automatic formalization of statements expressed in natural language is explored. Our proposal is a method using abstract representation and associating statistical and symbolic approaches.

Keywords

System Design, semantic parsing, abstract representation, formalization.

1 Introduction

La conception d'un système technique commence par un travail d'analyse des besoins et du contexte applicatif. Cette phase se traduit par la rédaction d'un cahier des charges, et nécessite de décrire toutes les caractéristiques fonctionnelles et comportementales du système. Ce document doit être précis, cohérent et complet. C'est une suite d'explications et de règles qui décrivent précisément toutes les propriétés du système à réaliser.
Les concepteurs peuvent être assistés dans leur tâche par des méthodes de spécification et de vérifications formelles, rigoureusement fondées et susceptibles de mettre en évidence des erreurs de conception au coût très élevé si non détectées. Elles offrent potentiellement un avantage indéniable. Pourtant, elles n'ont pas été largement adoptées. Cela est dû en partie à un certain nombre d'obstacles pragmatiques, tels que l'exigence d'une expertise appropriée, le besoin d'une formation spécifique ou l'absence de support adapté.
Il y a un écart important entre le bénéfice de ces méthodes en théorie, et leur intégration en pratique. Nos travaux visent à réduire cet écart.
Pour atteindre ce but, nous proposons un processus partant des documents produits par le concepteur, et rédigés dans une langue naturelle tel l'anglais. Nous souhaitons extraire de ces documents toutes les informations nécessaires à la construction de modèle formel. Partant d'énoncés informels, l'objectif est donc de produire un énoncé formel équivalent, et intégrable dans une démarche entièrement automatisée.
Le tableau 1 présente quelques exigences d'un cas d'étude spécifiant le fonctionnement d'un système d'accès, avec barrière de sécurité, pour un parking de voitures. Ce cas d'étude, étudié en détail du point de vue *"conception de système"* dans [3], a été implémenté avec l'outil Mica [6], une bibliothèque qui permet de définir des systèmes complexes en utilisant une syntaxe du langage OCaml. Dans cette mise en oeuvre, la formalisation des exigences a été réalisée manuellement (les exigences ont été traduites dans la syntaxe exigée par l'outil Mica). Notre objectif est d'enrichir ce genre d'outil pour pouvoir traiter directement les exigences exprimées en langue naturelle. Il s'agit de définir une brique qui pourrait y être reliée, un outil formalisant les énoncés dans un format exploitable pour automatiser la génération des modèles.

Id.	*Exigence*
E_1	When a ticket had been issued, the entry gate must open.
E_2	After an exit ticket is inserted, the exit gate must open.
E_3	If the parking is full or a vehicle is passing, the entry gate should not open.

TABLE 1 – Quelques exigences d'un système d'accès pour un parking de voitures

Nous nous intéressons en particulier aux règles comportementales décrivant les évolutions possibles ou interdites d'un système donné. Pour les besoins de nos expérimentations, nous avons défini un format précis de spécification sous la forme d'un triplet < M, P, C >. Ces spécifications

traduisent des règles associant un contexte donné C, une modalité M et une propriété P. Le contexte C est une formule logique portant sur un ensemble de propriétés du système, et permettant de cibler les états du système où la règle doit s'appliquer. Si ce contexte est vérifié, la modalité M est attribuée à la propriété P, définissant ainsi la manifestation possible, nécessaire ou interdite d'un évènement ou d'une propriété d'état. En associant une sémantique précise à cette représentation, et en combinant un ensemble de règles ainsi formalisées, il est envisageable de modéliser le comportement d'un système à différents niveaux d'abstraction.

Suivant l'état de l'art actuel, les techniques de traitement de langues naturelles ne peuvent garantir une traduction sans erreur, d'autant plus si l'on prend en compte le côté intrinsèquement ambiguë des langues naturelles. Nous pouvons cependant atténuer ce constat avec plusieurs observations sur les documents ciblés. En effet, ces documents sont généralement rédigés avec des formules répétitives, un style restreint et des formules qui se veulent explicites. La mise en évidence d'ambigüités serait donc déjà une aide bienvenue, et celles-ci apparaîtrait naturellement avec la construction de modèle incohérent. Les exigences du tableau E_1 et E_3 contiennent une certaine ambigüité ; elles sont en fait inconsistantes, et c'est justement ce type de contradictions qui sont visés.

Du point de vue traitement de la langue, les exigences du tableau 1 illustrent différents phénomènes linguistiques. La condition peut s'exprimer avec des mots clés variés (when, after, if) et une syntaxe différenciée. Ces énoncés traitent de propriétés, et la formalisation de ces énoncés implique d'extraire ces propriétés. Les spécifications contiennent aussi des modalités déontiques, formalisant l'obligation, l'interdiction, la permission et le facultatif. La formalisation d'un énoncé implique la prise en charge contrôlée de tous ces aspects. Plus précisément, il est nécessaire de différencier les entités (gate, ticket), les propriétés (open-gate, insert-ticket) et les groupes nominaux exprimant une relation entre ces éléments ou un phénomène linguistique.

Notre contribution associe deux niveaux d'analyse. Le premier niveau vise la construction de représentation abstraite générique à l'aide d'analyseur syntaxico-sémantique s'appuyant sur des méthodes statistiques. Ces représentations sont ensuite transformées en suivant des opérations de transduction, s'appuyant sur des règles symboliques, jusqu'à obtenir la définition formelle attendue.

La méthodologie proposée applique plusieurs étapes de transductions. Chacune de ces étapes préserve le sens de l'énoncé pour aboutir finalement à la représentation formelle attendue. Plusieurs opérations classiques sont généralement appliquées pour traiter des textes en langue naturelle, telles que le découpage en unités syntaxiques ou l'étiquetage grammatical. Nous appliquons ces opérations en pré-traitement d'une première phase d'analyse syntaxico-sémantique. Celle-ci permet de construire une structure abstraite capturant le sens de l'énoncé. Les représentations de sens abstrait (Abstract Meaning Representation, AMR), introduites par *Banarescu et al.* en 2013 [2], sont des structures intermédiaires adaptées pour cette étape. Nous exploitons en particulier deux algorithmes. Le premier met en oeuvre un processus itératif à partir d'analyse en dépendance (CAMR, [28]). Le second est fondé sur un modèle prédictif reliant des séquences à des graphes (STOG, [29]).

La deuxième phase du traitement consiste à transformer la représentation intermédiaire obtenue pour construire la représentation attendue. La structure abstraite de l'énoncé est analysée selon plusieurs angles, d'abord en s'appuyant sur un référentiel ontologique, puis suivant l'évaluation de requêtes logiques.

La méthodologie proposée a été implémentée et expérimentée sur deux cas d'études, le système d'accès pour un parking et le classique distributeur automatique. Ces deux cas d'études constituent deux corpus totalisant 50 phrases, et couvrant plusieurs phénomènes linguistiques (les conditions, les modalités, les connecteurs temporels et logiques et la négation). Le tableau 2 présente les représentations obtenues sur les exigences du tableau 1. Le taux de correction est de 75 % en moyenne sur les deux corpus expérimentaux.

Id.	*Exigence*
E_1	<necessary, open-gate, issue-ticket>
E_2	<necessary, open-gate, after insert-ticket>
E_3	<possible, not open-gate, full-parking or pass-vehicle>

TABLE 2 – Formalisation des exigences du tableau 1

2 Travaux connexes

Les méthodes formelles pour la conception de systèmes sont nombreuses. Nous pouvons citer les logiques temporelles, LTL et CTL [8, 7]. Les automates d'interfaces [9] et les interfaces modales [24, 5] visent à fournir une spécification fusionnée du comportement attendu et des environnements possibles. Une variante propositionnelle, les automates d'acceptation propositionnels [17], a été avancée récemment, améliorant l'expressivité pratique de ces méthodes. Les théories de contrats [3], avec une prise en charge native dans plusieurs langages de programmation [21], offrent un cadre flexible en tant que théorie générique.

L'intégration de ces systèmes se heurte à des difficultés pragmatiques que nous voulons dépasser. Notre contribution est un processus automatique assurant le passage de spécifications exprimées en langage naturel vers des spécifications formelles. La vocation de cette proposition est de fournir une interface entre des techniques formelles telles que présentées ci-dessus, et l'énoncé naturel de règles comportementales dans les documents d'ingénierie.

L'usage d'une représentation intermédiaire a été exploré dans plusieurs travaux, avec des approches variées. Une modélisation des connaissances par une ontologie peut être exploitée comme modèle de représentation pivot [26]. L'usage de patrons de spécifications de propriétés temporelles est une approche où le concepteur est guidé par

une grammaire structurée [11, 16]. Les langages contrôlés s'inscrivent dans la même logique. Le langage Attempto Controlled English (ACE, [13]) contraint la structure des phrases tout en restant très proche de l'anglais. Ce langage peut, par exemple, être utilisé pour traduire automatiquement des textes dans la logique du premier ordre.

Le projet ARSENAL [14] fournit un cadre méthodologique de spécifications logiques analysables (LTL). Le système Grammatical Framework [25] est une autre approche combinant langage source et langage cible. Basé sur les grammaires catégorielles et la théorie des types, ce système a été utilisé dans différents projets, et constitue une option intéressante si l'on se limite à un langage naturel contrôlé.

Nous posons la problématique un peu différemment. Là où les méthodes ci-dessus offrent un cadre pour construire une interface en langue naturelle pour différentes méthodes formelles, notre démarche vise l'analyse des documents pour en extraire les données nécessaires à l'utilisation entièrement automatisée d'une méthode formelle. Dans ce but, nous nous appuyons sur une représentation pivot. Cette idée est également présente, par exemple, dans des travaux visant la génération de requêtes SQL ou Sparql à partir de langue naturelle [23].

Les grammaires catégorielles combinatoires (CCG, [27]) sont intéressantes dans notre cas pour ses liens avec le lambda-calcul, qui pourrait être exploité comme représentation formelle intermédiaire. Les Grammaires Catégorielles Abstraites [10] en sont une généralisation traitant directement les catégories et le lambda-calcul. La théorie de représentation du discours (DRT, [15]), représentation d'un texte considérant le contenu sémantique qui dépend du contexte, peut être exploitée conjointement aux CCG pour augmenter la couverture des phénomènes linguistiques. Plusieurs analyseurs ont été proposés pour ces formalismes, tel que Ccg2Lambda [4]. Un système de résolution de problèmes mathématiques, acceptant la saisie en langage naturel, a ainsi été conçu en se basant sur les CCG [20].

Notre contribution vise la transformation d'un énoncé, en passant par une représentation sémantique intermédiaire, pour aboutir à une définition plus précise, manipulable et adaptable. Il s'agit d'un processus qui vise à compléter les approches mentionnées. Nous avons fait le choix d'utiliser les représentations de sens abstrait (Abstract Meaning Representation, AMR], un formalisme basé sur les graphes et utilisé pour capturer la sémantique au niveau des phrases [2]. Notre approche pourrait aussi bien être associée au CCG, à l'image des travaux de *Matsuzaki et al* [20]. Elle en est une alternative, dans un contexte différent (la conception de systèmes), mettant l'accent sur les modalités.

Un premier analyseur AMR, introduit par *Flanigan et al* en 2014 [12], produit une analyse syntaxique en se basant sur l'alignement entre mots et concepts. L'identification des concepts est d'abord traitée comme une tâche d'étiquetage de séquence. Les relations sont ensuite identifiées en s'appuyant sur des graphes d'analyse de dépendance. Une approche similaire a été utilisée par *Zhou et al* [30], où l'identification des concepts et des relations est réalisée conjointement, et *Liu et al* [18], où les alignements sont utilisés comme des variables latentes d'un modèle probabiliste conjoint, avec une amélioration substantielle des résultats obtenus.

L'algorithme CAMR [28] transforme progressivement des analyses de dépendance en AMR, suivant un processus itératif sur les noeuds des arbres de dépendance. D'autres travaux s'appuient également sur cette approche, avec des résultats comparables.

Une méthode inductive, basée sur les grammaires CCG, a également été proposée en exploitant des ressources sémantiques pour convertir des formules logiques en AMR [1]. Plus récemment, un algorithme de prédiction a obtenu de très bons résultats avec un modèle basé attention, reliant séquences et graphes [29].

L'utilisation de ces analyseurs permet de construire des représentations formelles qu'il s'agit maintenant d'exploiter pour obtenir des définitions formelles précises, exploitables avec différentes méthodes de modélisation. C'est ce que nous allons voir dans la prochaine section.

3 Analyse sémantique

Structure intermédiaire entre l'expression naturelle d'un énoncé et leurs traitements par des méthodes automatiques, les représentations sémantiques permettent de capturer formellement la signification d'un texte. L'analyse sémantique proposée ici exploite ces structures. Elle est constituée de plusieurs grandes phases (figure 1). L'idée générale est de partir d'une représentation sémantique intermédiaire, obtenue après une analyse syntaxico-sémantique de l'énoncé initial, puis de transformer cette représentation pour aboutir à la définition formelle attendue.

Premièrement, les données à traiter sont regroupées dans un unique fichier. Les opérations de pré-traitements sont classiques. Elles comprennent le découpage des éléments de chaque phrase en unité syntaxique (token) et l'étiquetage grammatical (POS tagging). Toutes ces opérations sont réalisées avec l'outil Stanford CoreNLP [19].

Dans un second temps, une étape importante du processus consiste en une analyse précise de l'énoncé pour construire une structure formelle qui en capture le sens. Plusieurs travaux récents étudient la construction de telles représentations sémantiques, comme développés dans la section précédente. Nous avons fait le choix d'utiliser les représentations de sens abstrait (Abstract Meaning Representation, AMR], un formalisme basé sur les graphes et utilisé pour capturer la sémantique au niveau des phrases [2].

La construction de cette structure formelle intermédiaire n'est pas une fin en soi, mais un moyen pour aboutir à la définition formelle attendue. L'autre étape importante du processus constitue notre contribution majeure. Elle vise la transformation de la structure intermédiaire dans une démarche contrôlée.

La dernière phase du traitement permet de finaliser le travail en appliquant différentes opérations de finitions sur l'énoncé obtenu. Cette étape n'est pas fondamentale et dé-

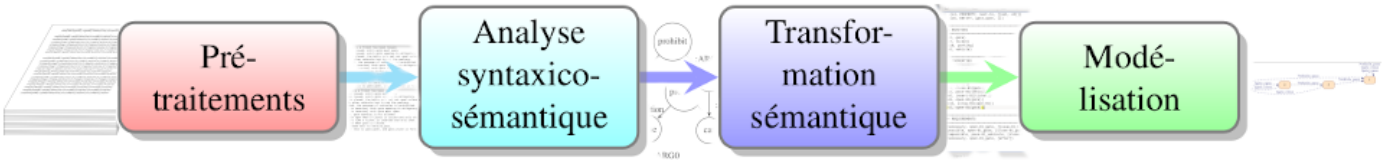

FIGURE 1 – Chaine de traitements

pend fortement du format de l'énoncé à produire, et de son usage.
Nous avons ainsi défini une chaine de traitement complète, modulaire et intégrable avec différentes méthodes formelles. Cette brique n'a pas d'intérêt en elle-même. Bien intégrée, en revanche, elle est susceptible de rendre exploitable, en pratique, de nombreux outils ou représentations formelles.
Dans la suite de cette section, nous détaillons les deux phases principales de la chaine de traitement proposée, la phase d'analyse syntaxico-sémantique, mise en oeuvre pour l'AMR dans nos expérimentations, et la phase de transduction sémantique.

3.1 Analyse syntaxico-sémantique pour produire une représentation intermédiaire

Les représentations de sens abstrait (Abstract Meaning Representation, AMR) permettent de capturer la sémantique au niveau des phrases. Ce formalisme prend la forme d'un graphe où les noeuds correspondent aux concepts présents dans la phrase. Les arêtes orientées reliant les noeuds du graphe traduisent des relations entre ces concepts. Cette structure fournit ainsi une représentation formelle exploitable préservant le sens de l'énoncé.
La figure 2 représente l'AMR de l'énoncé E_1 (voir tableau 1).

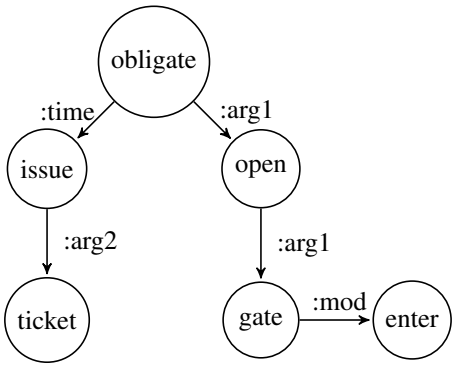

FIGURE 2 – Abstract Meaning Representation (AMR) pour l'énoncé E_1.

L'analyse syntaxico-sémantique pour AMR est une tâche qui consiste à transformer un énoncé textuel, écrit dans un langage naturel, en une représentation de sens abstrait. Notre outil intègre deux algorithmes [28, 29]. Le premier consiste à transformer progressivement des analyses de dépendance en AMR, avec un processus itératif sur les noeuds des arbres de dépendance. Le second est plus récent et performant. C'est un algorithme de prédiction utilisant un modèle basé attention, reliant séquences et graphes.
A l'issue de cette étape, l'énoncé a été transformé en une représentation formelle qu'il nous faut maintenant exploiter pour aboutir à la représentation voulue.

3.2 Transduction sémantique

Le processus de transduction sémantique consiste à passer d'une représentation formelle intermédiaire à la représentation cible. La première représentation est obtenue lors de l'étape précédente à partir d'une analyse syntaxico-sémantique de l'énoncé initial. Il s'agit maintenant de transformer la structure obtenue.
L'algorithme proposé comprend plusieurs étapes :

- analyse de la structure abstraite (AMR) pour produire un arbre de syntaxe abstraite (AST)
- analyse de l'AST pour extraire des "données abstraites", en s'appuyant sur une table des symboles et un classificateur
- analyse des données abstraites pour définir une sémantique abstraite sous la forme de trois ensembles (concepts, instances et valuation)
- analyse de la définition abstraite sur une liste de motifs de transduction, associant une requête logique et un modèle pour la représentation à produire

Analyse syntaxique de la structure intermédiaire. La structure abstraite intermédiaire, obtenue en sortie de la phase précédente, constitue la source entrante de la phase de transduction. Nous proposons d'utiliser les techniques classiques mises en oeuvres dans les compilateurs pour analyser cette représentation. En particulier, une définition de syntaxe abstraite (ASD) est proposée pour l'AMR. Celle-ci permet de produire l'arbre de syntaxe abstraite (AST) correspondant à l'énoncé en entrée, et toute une série d'éléments de calcul (table des symboles et données abstraites). La figure 3 montre pour exemple la table des symboles de l'énoncé E_1. Chaque symbole est associé à une catégorie évaluée à l'aide du classificateur.

Classificateur. Les instances et les relations entre ces instances sont analysées avec un module nommé classificateur. Ce module permet d'attribuer une catégorie à chaque instance, et ainsi de différencier les instances qui couvrent un phénomène linguistique de celles qui définissent une entité (instance d'un concept élémentaire) ou une propriété

```
<corpus_parking_E1> Symbol Table:
(vv1, obligate-01,
  [(ARG2, vv2), (time, vv5)],
  MODALITY)
(vv3, gate, [(mod, vv4)], ENTITY)
(vv2, open-01, [(ARG1, vv3)], PROPERTY)
(vv5, issue-01, [(ARG1, vv6)],
  PROPERTY)
(vv4, enter-01, [], PROPERTY)
(vv6, ticket, [], ENTITY)
```

FIGURE 3 – Table des symboles de l'énoncé E_1.

(association de concepts). Il s'appuie sur une ontologie regroupant les concepts dans des ensembles structurés, et basé sur un corpus de référence. Nous utilisons la PropBank [22], un corpus annoté de propositions verbales et de leurs arguments. Le choix de la PropBank comme référenciel est directement lié au choix de la représentation intermédiaire, l'AMR s'appuyant également sur ce corpus. L'usage d'un autre référentiel est possible, impliquant néanmoins un alignement entre les concepts de la PropBank, utilisés pour l'AMR, et ceux du corpus choisi.

Définition d'une sémantique abstraite. La définition de la sémantique abstraite définit plusieurs ensembles caractérisant complètement la signification de l'énoncé traité. Il s'agit de l'ensemble de concepts, des instances (associées à différents attributs utiles) et des prédicats vérifiés pour l'énoncé. L'ensemble des prédicats vérifiés définit une valuation de l'énoncé susceptible de vérifier une formule logique. Un exemple est donné, figure 4, pour l'énoncé E_1.

Application des motifs de transduction. Les motifs de transduction associent une requête logique, et un modèle utilisé pour produire l'énoncé sous la forme attendue. Ils constituent des paramètres du système de transduction. Comme illustré dans le tableau 3, les requêtes sont des formules exprimées dans une forme restreinte de la logique de premier ordre, utilisant uniquement l'opérateur de conjonction logique. Elles permettent de capturer les énoncés qui correspondent à la forme recherchée. Les modèles définissent le format des énoncés à produire en prenant en compte les caractéristiques associées aux instances. De fait, il est possible de modifier la forme des représentations en sortie en modifiant les modèles associés aux requêtes. La définition sémantique de l'énoncé est évaluée pour chaque motif. Si la requête est vérifiée, la sortie est générée en appliquant le modèle aux instances validant la requête.

Q.	$\lambda x.\lambda y.\lambda z.isModality(x) \wedge isProperty(y) \wedge \ldots$ $\wedge arg+(x,y) \wedge time(x,z) \wedge \ldots$
T.	$< [1].modality, [2].property, [3].property >$

TABLE 3 – Exemple d'un motif de transduction associant une requête (Query) et un modèle (Template).

4 Expérimentations

La chaine de traitement proposée a été complètement implémentée avec les langages python et java. Le module principal, écrit en python, permet de contrôler les différentes opérations. Stanford CoreNLP, un ensemble d'outils de traitement du langage naturel, est utilisé pour le prétraitement. L'analyse syntaxico-sémantique pour la génération des structures AMR peut être utilisé en choisissant l'un des deux algorithmes proposés. La mise en oeuvre de ces algorithmes est une adaptation des outils CAMR [28] et STOG [29], dont le code est sous licence libre. Le module ATP (Abstract Transduction Process) est la contribution majeure de notre implémentation. Ce module a été developpé en java. Il intègre toutes les étapes de transduction sémantique présentées en section 3.2. Le paramétrage de l'outil comprend la définition de l'ontologie (module OntoMap), utilisée par le classificateur, et la définition des motifs de transduction (module TransductionPattern), utilisée lors de l'étape d'analyse sémantique abstraite. En l'état actuel, seul un fragment du corpus PropBank [22] est intégré dans l'ontologie, fragment suffisant pour couvrir quelques phénomènes linguistiques.

L'expérimentation a été menée sur deux cas d'études, le système d'accès pour un parking de voitures, et le classique distributeur automatique. Ces deux cas d'études forment deux corpus, pour un total de 50 phrases. Ils couvrent plusieurs phénomènes linguistiques tels que les conditions, les modalités, la négation ou encore les connecteurs temporels et logiques.

Les résultats obtenus sont rassemblés dans le tableau 4. La dimension (colonne Dim.) des corpus est précisée. Le résultat 1 (colonne Rés. 1) est le taux de correction observé après la première phase du traitement, tandis que le résultat 2 (colonne Rés. 2) correspond au taux de correction final, évalué après la seconde phase.

Corpus	*Dim.*	*Rés. 1*	*Rés. 2*
Parking Gates	30	0.85	0.85
Vending Machine	20	0.87	0.67
TOTAL (moyenne)	50	0.87	**O.77**

TABLE 4 – Résultats obtenus sur les corpus "Parking Gates" et "Vending Machine".

Les exigences du premier corpus décrivent le fonctionnement d'un système d'accès, avec barrière de sécurité, pour un parking de voitures. Ce cas d'étude a été étudié en détail dans [3] du point de vue *"conception de système"*. Le corpus constitué couvre les phénomènes linguistiques ciblés suivant des proportions inégales. Les modalités sont présentes dans 90% des phrases. Les règles sont décrites avec une condition (when, if) dans 73% des cas et avec un connecteur temporel (after, before) pour 27% des phrases. Ce corpus contient également des connecteurs logiques (10%), des négations (20%) et des adverbes (10%).

Le second corpus est constitué de 20 phrases décrivant un système de distribution automatique. Il couvre les

```
<corpus_parking_E1> Abstract Semantic:
concepts = [open-01, issue-01, ticket, obligate-01, gate, enter-01]
instances = [vv1, vv3, vv2, vv5, vv4, vv6]
predicates = [isModality(vv1), isProperty(vv5), mod(vv3, vv4), isEntity(vv3),
  isProperty(vv4), arg+(vv1, vv2), isProperty(vv2), arg1(vv2, vv3), time(vv1, vv5),
  arg+(vv5, vv6), isEntity(vv6), arg2(vv1, vv2), arg1(vv5, vv6), arg+(vv2, vv3)]
```

FIGURE 4 – Définition d'une sémantique abstraite pour l'énoncé E_1. Les instances sont associées à plusieurs caractéristiques non représentées ici. Les prédicats peuvent être évalués par des requêtes logiques.

mêmes phénomènes linguistiques avec des proportions différentes : 20% pour les conditions, 65% pour les modalités, 55% pour les connecteurs temporels, 15% pour les connecteurs logiques et 35% pour les négations. Il contient également 25% de phrases affirmatives simples n'ayant pas de représentation attendue. La présence de ces phrases montre que la méthode proposée permet de trier automatiquement les données en entrée en ne générant une représentation que lorsque celle-ci a du sens.

5 Conclusion

Cet article s'inscrit dans une progression autour d'un axe visant la formalisation efficace d'énoncés exprimés en langue naturelle. Une première présentation en a été fait en atelier, avec quelques uns des principes envisagés et une implémentation sommaire pour soutenir nos intuitions originelles.

Une deuxième étape est validée ici, importante, avec une proposition mieux structurée, une implémentation stable et une expérimentation sur deux corpus certes limités, mais néanmoins complets. Nous avons ainsi montré que la méthode proposée fonctionne, en couvrant plusieurs phénomènes linguistiques avec une certaine efficacité, et avec également un potentiel d'extension.

L'expérimentation menée est néanmoins insuffisante. Nous couvrons quelques phénomènes linguistiques, partiellement. Il est possible d'améliorer le niveau de couverture en enrichissant le référentiel ontologique. De même, le paramétrage des motifs de transduction (requêtes et modèles) peut être complété. Il est également possible, théoriquement, d'appliquer cette méthode pour construire d'autres représentations. Par ailleurs, la méthodologie proposée peut s'appuyer sur une autre représentation intermédiaire, même si ce changement de structure demanderait un effort plus important.

La suite de nos travaux devra permettre de montrer la validité de ces différents points, en menant des expérimentations plus complètes. Au delà, la généralisation de notre démarche pourrait inclure l'apprentissage des motifs de transduction, à partir de quelques exemples positifs, et offrir ainsi la promesse d'un outil plus complet.

Références

[1] Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710. Association for Computational Linguistics, Lisbon, Portugal, September 2015.

[2] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics, Sofia, Bulgaria, 2013.

[3] Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger, and Kim G. Larsen. *Contracts for System Design*. 2018.

[4] Johannes Bos. Open-domain semantic parsing with boxer. *Proceedings of the 20th Nordic Conference of Computational Linguistics*, pages 301–304, 2015.

[5] Ferenc Bujtor, Sascha Fendrich, Gerald Lüttgen, and Walter Vogler. Nondeterministic modal interfaces. In *SOFSEM 2015 : Theory and Practice of Computer Science*, pages 152–163. Springer Berlin Heidelberg, 2015.

[6] Benoît Caillaud. Mica : A Modal Interface Compositional Analysis Library, October 2011.

[7] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. volume 8, pages 244–263. ACM, New York, NY, USA, 1986.

[8] Edmund M Clarke and E Allen Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logic of programs : Workshop*, volume 131, pages 244–263. 1981.

[9] Luca de Alfaro and Thomas A. Henzinger. Interface automata. *SIGSOFT Softw. Eng. Notes*, 26(5) :109–120, September 2001.

[10] Philippe de Groote. Towards abstract categorial grammars. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 252–259. Association for Computational Linguistics, 2001.

[11] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 411–420, New York, NY, USA, 1999. ACM.

[12] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1426–1436. Association for Computational Linguistics, Baltimore, Maryland, 2014.

[13] Norbert E. Fuchs and Rolf Schwitter. Specifying logic programs in controlled natural language. *CoRR*, abs/cmp-lg/9507009, 1995.

[14] Shalini Ghosh, Daniel Elenius, Wenchao Li, Patrick Lincoln, Natarajan Shankar, and Wilfried Steiner. abs/1403.3142, 2014.

[15] Hans Kamp. A theory of truth and semantic representation. In J. A. G. Groenendijk, T. M. V. Janssen, and M. B. J. Stokhof, editors, *Formal Methods in the Study of Language*, volume 1, pages 277–322. Mathematisch Centrum, Amsterdam, 1981.

[16] Sascha Konrad and Betty H. C. Cheng. Real-time specification patterns. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 372–381, New York, NY, USA, 2005. ACM.

[17] Aurélien Lamercerie. Une algèbre des automates d'acceptation propositionnelle déterministes comme théorie d'interface pour la conception de systèmes cyberphysiques, 2019. Poster.

[18] Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. An amr aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, page 2422–2430. Association for Computational Linguistics, Brussels, Belgium, 2018.

[19] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, pages 55–60. Association for Computational Linguistics, Baltimore, Maryland, June 2014.

[20] Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H. Arai. Semantic parsing of pre-university math problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 2131–2141. Association for Computational Linguistics, Vancouver, Canada, July 2017.

[21] Bertrand Meyer. *Touch of Class : Learning to Program Well Using Object Technology and Design by Contract*. Springer, Software Engineering, 2009.

[22] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank : An annotated corpus of semantic roles. *Computational Linguistics*, 31(1) :71–106, 2005.

[23] Camille Pradel, Ollivier Haemmerlé, and Nathalie Hernandez. Natural language query interpretation into SPARQL using patterns. In *Fourth International Workshop on Consuming Linked Data - COLD 2013*, pages pp. 1–12, Sydney, Australia, October 2013.

[24] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, Axel Legay, and Roberto Passerone. A modal interface theory for component-based design. *Fundamenta Informaticae*, 108(1-2) :119–149, 2010.

[25] Aarne Ranta. *Grammatical Framework : Programming with Multilingual Grammars*. CSLI Publications, Stanford, 2011. ISBN-10 : 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

[26] Driss Sadoun, Catherine Dubois, Yacine Ghamri-Doudane, and Brigitte Grau. From natural language requirements to formal specification using an ontology. In *IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI 2013)*, pages 755–760, Herndon, VA, United States, 2013.

[27] Mark Steedman. *The syntactic process*. MIT Press, 2000.

[28] Chuan Wang, Nianwen Xue, and Sameer Pradhan. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 366–375. Association for Computational Linguistics, Denver, Colorado, 2015.

[29] Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. Amr parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94. Association for Computational Linguistics, Florence, Italy, 2019.

[30] Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689. Association for Computational Linguistics, Austin, Texas, 2016.