ARES : un extracteur d'exigences pour la modélisation de systèmes

Aurélien Lamercerie

ARES : un extracteur d'exigences pour la modélisation de systèmes

Aurélien Lamercerie*

*Univ Rennes, Inria, IRISA - UMR 6074, F-35000 Rennes, France
aurelien.lamercerie@inria.fr

Résumé. L'application de méthodes formelles pour assister la conception de systèmes s'appuie sur une modélisation des comportements attendus. La construction de ces représentations nécessite d'extraire les règles comportementales (exigences) généralement définies dans un document de spécifications. Le logiciel ARES (Abstract Requirement Extraction for Systems) répond à ce besoin en partant d'énoncés en langage naturel. Cet outil exploite une représentation sémantique intermédiaire (AMR), et permet de construire un ensemble de définitions abstraites directement exploitables pour modéliser le comportement de systèmes.

1 Introduction

Les représentations sémantiques constituent un intermédiaire intéressant entre l'expression naturelle d'un énoncé et leurs traitements par des méthodes automatiques. Elles permettent de capturer formellement la signification d'un énoncé, le rendant plus accessible pour un traitement automatique. Nous proposons d'exploiter ce type de représentations pour analyser des documents techniques et en extraire les règles comportementales décrivant les évolutions possibles ou interdites d'un système donné.

Nous considérons en particulier des systèmes techniques tels que les avions, les centrales électriques, les véhicules autonomes, etc. Nous nous intéressons plus particulièrement à l'aspect comportemental des systèmes étudiés, spécifié sous la forme d'un ensemble de règles appelées exigences. Par exemple, le cahier des charges d'un système d'accès pour un parking de voitures pourrait contenir des règles telles que les exigences R_1 et R_2 :

— ($R1$) The passage of a car is prohibited if the security gate is closed [1].

— ($R2$) Entry gate must open once a ticket has been issued [2].

L'objectif serait d'assister la conception de tels systèmes dès les premières étapes du cycle de développement. Cela nécessite l'extraction des exigences définies et la construction de représentation formelle. Le logiciel ARES (Abstract Requirement Extraction for Systems) répond à ce besoin. Il permet d'extraire les exigences contenues dans un ou plusieurs documents et fournit en sortie un ensemble de définitions abstraites directement exploitables pour modéliser le comportement de systèmes. Il intègre un analyseur syntaxico-sémantique pour des

1. Le passage d'une voiture est interdit si la barrière de sécurité est en position fermée.
2. La barrière d'entrée doit s'ouvrir une fois le ticket émis.

représentations de sens abstrait (Abstract Meaning Representation, AMR), formalisme qui a fait l'objet de plusieurs travaux récents prometteurs (Banarescu et al. (2013)). La transformation de ces représentations permet de construire des ensembles de définitions abstraites de plus haut niveau, en différenciant les entités du système, les propriétés à évaluer et les exigences à respecter.

2 Chaîne de traitements

Nous cherchons à résoudre un problème, celui de la modélisation du comportement de système, en partant des données telles qu'elles sont disponibles en pratique. Elles prennent généralement la forme de cahier des charges regroupant un ensemble d'exigences, exprimées en langage naturel. La figure 1 donne une vue générale de la chaine de traitements proposée.

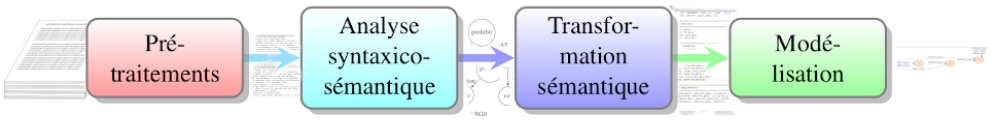

FIG. 1 – *Chaine de traitements du logiciel ARES*

Le processus mis en oeuvre n'implique aucune contrainte particulière sur les données en entrée, que ce soit sur la forme du document ou le texte. La phase de pré-traitement permet d'aboutir à un découpage standard dont l'unité est la phrase. Cette étape, qui n'est pas automatisé dans la version actuelle de l'outil, aboutit à la construction d'une liste d'exigences potentielles. La suite du traitement retiendra les phrases qui correspondent à l'un des motifs sémantiques définis dans l'outil, ces motifs déterminant la structure sémantique attendue des exigences à extraire.

Le traitement principal est entièrement automatisé. Il consiste à produire une représentation abstraite pour chaque exigence, et à transformer chacune de ces représentations en suivant des motifs sémantiques dont la définition est une partie intégrante de l'outil. Le résultat est une construction formelle associant un contexte et une propriété, et caractérisant une évolution possible, interdite ou obligatoire du système.

Les représentations formelles obtenues peuvent être exploitées pour mettre en évidence des erreurs de conception en proposant une modélisation du système étudié. Notre outil intègre une implémentation des interfaces modales sous la forme d'une librairie O'CAML (Caillaud (2011)). Ces modèles supportent des méthodes de raisonnement compositionnel sur les systèmes réactifs, applicables notamment dans l'ingénierie des systèmes, en prenant en compte la variabilité des réalisations possibles induites par une exigence.

2.1 Analyse syntaxico-sémantique

L'objectif d'un analyseur syntaxico-sémantique pour AMR est de construire la représentation abstraite (AMR) d'un énoncé exprimé dans un langage naturel. Plusieurs travaux ont été

publiés ces dernières années avec des approches variées. Le score Smatch, proposé par Cai et Knight (2013), est utilisé pour apprécier les résultats obtenus. Il permet d'évaluer la distance sémantique entre deux représentations, et peut être utilisé pour estimer la qualité d'une représentation construite par une méthode automatique à celle d'un corpus de référence. Les meilleurs parseurs AMR actuels, tels que Liu et al. (2018) ou Zhang et al. (2019) par exemple, obtiennent des scores autour de 70 % sur les corpus de référence.

Notre logiciel intègre le parseur syntaxico-sémantique CAMR, décrit par Wang et al. (2015). La méthodologie proposée consiste à transformer progressivement des analyses de dépendance en AMR, avec un processus itératif sur les noeuds des arbres de dépendance. Ce parseur permet de construire une représentation abstraite pour chaque exigence traitée, comme illustré sur la figure 2 qui montre l'AMR obtenu pour l'exigence $R1$. Sur ce graphe, les arcs définissent des relations entre concepts, chaque arc étant étiqueté par le rôle associé (:ARG0 et :ARG1 sont des rôles centraux, :condition est un rôle spécifique).

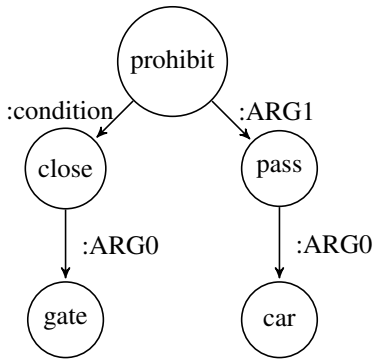

FIG. 2 – *AMR correspondant à l'exigence $R1$*

2.2 Transformation sémantique

La première phase de notre processus produit une représentation abstraite sous forme d'AMR pour chaque exigence potentielle. La seconde étape consiste à transformer ces représentations pour aboutir à un ensemble de définitions abstraites de plus haut niveau, directement exploitables pour la modélisation de systèmes. En sortie, trois ensembles sont proposés : un ensemble d'entités, un ensemble de propriétés et un ensemble d'exigences.

La distinction entre les entités et les propriétés est fondée sur la nature des concepts présents dans la représentation étudiée. Les concepts qui correspondent à une modalité, déontique ou temporelle, ou un opérateur logique (et, ou), sont écartés dans un premier temps. Ils sont exploités pour la définition de l'exigence. Les concepts restants sont considérés comme entité s'ils ne sont pas fondamentalement reliés à d'autres concepts, c'est à dire si aucun rôle de base (core role) ne les relie à un autre concept. A l'inverse, les ensembles de concepts fondamentalement connectés définissent des propriétés.

La définition abstraite d'une exigence est une relation entre un contexte et une propriété. Cette définition peut se représenter comme une AMR de plus haut niveau, où un concept modal

est relié à une évolution envisagée (propriété) et un contexte (ensemble de propriétés). Différents motifs sémantiques sont associés aux définitions abstraites attendues pour déterminer la structure sémantique des exigences à extraire.

A l'issue de ces deux phases, l'analyse de l'exigence $R1$ interdisant le passage d'un véhicule si la barrière est fermée permet de produire un ensemble d'entités, contenant les entités "security-gate" et "vehicle", un ensemble de propriétés, avec les propriétés "security-gate-close" et "vehicle-pass", et la définition "(Interdiction, vehicle-pass, security-gate-close)".

3 Expérimentations et perspectives

Une première expérimentation a été réalisée en s'appuyant sur quelques systèmes théoriques, et donne une indication positive sur l'approche proposée, à confirmer sur un corpus plus complet.

Une analyse préalable a été nécessaire pour adapter manuellement les données en entrée, et construire une liste d'exigences potentielles. L'automatisation de cette étape et différents traitements préliminaires (classification, reconnaissance d'entités nommées) peuvent être envisagée pour améliorer le processus.

De même, plusieurs opérations pourrait être utiles pour ajuster le résultat obtenu, ou classer les exigences. Il est possible qu'une même entité ou propriété soit désignée par des concepts sémantiques différents. Une phase d'alignement, s'appuyant sur une ontologie, permettrait de corriger ces écarts en définissant une équivalence entre différents ensembles de concepts sémantiques.

Références

Banarescu, L. et al. (2013). Abstract meaning representation for sembanking.

Cai, S. et K. Knight (2013). Smatch : an evaluation metric for semantic feature structures.

Caillaud, B. (2011). Mica : A Modal Interface Compositional Analysis Library.

Liu, Y. et al. (2018). An amr aligner tuned by transition-based parser.

Wang, C., N. Xue, et S. Pradhan (2015). A transition-based algorithm for AMR parsing.

Zhang, S. et al. (2019). Amr parsing as sequence-to-graph transduction.

Summary

Using formal methods to assist system design relies on expected behaviors modeling. The construction of these representations requires extracting behavior rules, called requirements, generally defined in a specification document. ARES (Abstract Requirement Extraction for Systems) meets this need starting from a statement of requirements in natural language. This tool operates an intermediate semantic representation (AMR), and converts it into exploitable formal requirements to model the behaviors of systems.