Expérimentation pour la définition d'une sémantique dans l'IDM

Benoît Combemale — Xavier Crégut
Pierre-Loïc Garoche — Xavier Thirioux

FéRIA - IRIT-ACADIE, INPT-ENSEEIHT
2, rue Charles Camichel, BP 7122
F-31071 Toulouse Cedex 7
prénom.nom@enseeiht.fr

RÉSUMÉ. *Notre objectif est de définir une sémantique pour un DSL donné dans le but soit de pouvoir le simuler et donc animer ses modèles, soit de vérifier des propriétés sur ses modèles, en utilisant par exemple des techniques de type model-checking. Dans ces deux cas, ce que l'on formalise c'est la sémantique du DSL qui est nécessairement connue de son concepteur, même si elle est souvent informelle. Ayant mené plusieurs expérimentations pour définir une sémantique opérationnelle d'une part, et une sémantique par traduction d'autre part, nous précisons dans quels cas ces sémantiques nous paraissent judicieuses. Nous proposons aussi une approche outillée permettant d'allier vérification et simulation, et donc sémantique par traduction et sémantique opérationnelle.*

ABSTRACT. *Our objective is to define the semantics for a given DSL either to simulate its models or to check properties on them using model-checking techniques. In both cases, the purpose is to formalize the DSL semantics that is known by the DSL Designer, often in an informal way. After several experiments to define operational semantics on the one hand, and a translationnal semantics of the other hand, we specify in which cases these semantics seem to be judicious. We propose also an approach allowing to combine verification and simulation, and thus translational semantics and operational semantics.*

MOTS-CLÉS : *IDM, sémantique opérationnelle, sémantique de traduction*

KEYWORDS: *MDE, operational semantics, translationnal semantics*

1. Introduction

Dans l'IDM (*Ingénierie Dirigée par les Modèles*), les modèles sont définis par des méta-modèles qui en spécifient la syntaxe et certaines propriétés structurelles, généralement sous la forme d'une syntaxe abstraite (MOF, Ecore, KM3, etc) complétée de contraintes exprimées dans un langage de requête tel qu'OCL. L'IDM favorise également la définition de langages dédiés (*Domain Specific Language*) qui ont l'avantage de permettre aux utilisateurs de se concentrer sur leur métier en manipulant un formalisme spécifique à leur activité. Pour cela, de nombreux ateliers (Topcased, GME, AMMA...) permettent de définir facilement la syntaxe abstraite et concrète d'un DSL.

À ce jour, un point fort de la recherche est l'expression d'une sémantique comportementale permettant l'exécution des modèles construits au cours d'un développement. Les travaux que nous menons dans ce contexte s'inspirent des différentes formes d'expression de la sémantique comportementale définies dans l'ingénierie des langages de programmation. Nous avons mené des expérimentations pour définir une sémantique opérationnelle en utilisant un langage de méta-programmation (p.ex. Kermeta (Muller *et al.,* 2005)) ou des transformations endogènes (exprimées en ATL (Jouault *et al.,* 2005), ou en AGG (Taentzer, 2003), un outil de réécriture de graphe) et comparé ces approches dans (Combemale *et al.,* 2006). Nous avons pu dans les deux cas simuler un modèle mais un enrichissement du méta-modèle était nécessaire pour capturer l'état global du système. La principale différence est qu'une approche par méta-programmation consiste à enrichir le méta-modèle d'opérations équipées d'une implantation qui, dans une approche par transformations endogènes, sont décrites en dehors du méta-modèle, dans la transformation elle-même.

Nous avons expérimenté la définition d'une sémantique de traduction (Combemale *et al.,* 2007) vers les réseaux de Petri à l'aide de transformations décrites en ATL. Les réseaux de Petri étant exécutables, on définit ainsi une sémantique d'exécution pour un modèle de DSL. Cette technique permet ainsi de réutiliser les outils (p.ex. model checker, simulateur...) fournis par le domaine cible mais nécessite d'interpréter les résultats obtenus par rapport aux modèles de départ.

Suite à ces différentes expériences, nous proposons dans cet article une synthèse sur la définition d'une sémantique dans le cadre d'un développement dirigée par les modèles. Pour cela, la section 2 compare sémantique opérationnelle et sémantique de traduction. Dans la section 3 nous indiquons comment peut être exprimée une sémantique, en particulier son niveau d'abstraction et la manière dont elle est décrite, puis nous indiquons comment, dans le cas d'une sémantique de traduction, la cohérence de la traduction peut être vérifiée. Enfin, dans la section 4, nous présentons une approche combinant vérification (par une sémantique de traduction) et simulation (par une sémantique opérationnelle).

2. Sémantique opérationnelle Vs. Sémantique de traduction

Une sémantique opérationnelle permet de rester dans le même espace technologique (Favre *et al.,* 2006) et d'exprimer les changements d'états des modèles dans

le même cadre métier. À l'inverse, une sémantique par traduction met en œuvre une transformation vers un autre espace technologique formellement défini (Clark *et al.,* 2004). Cette dernière approche permet ainsi de bénéficier de tous les outils disponibles dans l'espace cible mais nécessite d'interpréter l'exécution d'un modèle du langage de départ en fonction de la sémantique d'exécution des concepts du langage cible. D'autre part, il se pose le problème de la remontée des résultats obtenus vers l'espace d'origine.

Ainsi, une sémantique opérationnelle nous semble plus simple à mettre en œuvre, en particulier parce que, restant dans le même espace technologique, les concepts manipulés sont plus proches du domaine métier. Dans un objectif d'animation (visualisation de l'évolution d'un système au cours de son exécution) et/ou de simulation (interprétation d'un scénario, c-à-d ensemble ordonné d'évènements, guidant une exécution), une telle approche nous semble préférable, en particulier si le modèle de calcul pour la simulation est simple (p.ex. événements discrets).

Toutefois, si l'on souhaite recourir à des techniques de model checking, une approche par traduction est plus adaptée pour tirer parti des outils de vérification disponibles dans l'espace technologique cible. Il est alors nécessaire de définir un métamodèle du langage cible, une traduction et, enfin, un extracteur en fonction de la syntaxe concrète attendue par les outils visés[1].

Nous affinons dans le paragraphe suivant les types de sémantiques possibles du langage de départ dans le cadre de l'expression d'une traduction.

3. Taxonomie pour la définition d'une traduction

Nous énumérons ici les différentes approches permettant de décrire la sémantique d'un modèle via la traduction vers un modèle cible à la sémantique bien définie. D'une manière générale, nous considérerons dans la suite que le modèle cible de la traduction est un modèle muni d'une sémantique formelle opérationnelle à petit-pas, telle les réseaux de Petri. La sémantique du DSL source peut être décrite de deux points de vue orthogonaux.

3.1. *Expression de la sémantique du DSL source*

Le premier est celui du **niveau d'abstraction** de la sémantique. Une sémantique opérationnelle à petits pas peut, par exemple, être décrite par un ensemble de règles de réécriture, par un automate ou une structure de Kripke. Une abstraction dénotationnelle de cette sémantique, associera à chaque état ses images possibles par toutes les applications possibles de ces transitions ou règles. Enfin une sémantique axiomatique, abstraction de ces deux sémantiques, n'est pas opératoire et consiste en un ensemble de propriétés satisfaites par le modèle, sans décrire complètement son comportement

1. Nous proposons une étude de cas complète pour la définition d'une sémantique de traduction à l'adresse : http://www.eclipse.org/m2m/atl/usecases/SimplePDL2Tina/

(Winskel, 1993). Dans la suite, nous distinguerons deux types de précisions : les sémantiques opératoires, comme une sémantique à petit pas, des sémantiques de spécification, dénotationelles ou axiomatiques, qui permettent de décrire partiellement le comportement du modèle.

Le second point de vue, très pragmatique, est celui du **moyen de description** de cette sémantique. La sémantique, quelque soit son niveau d'abstraction, peut être décrite de façon formelle par une structure de Kripke, des règles de réécriture, ou des transformations endogènes, en ATL par exemple. Les DSL, étant initialement utilisés par les concepteurs pour communiquer leurs concepts de modélisation, n'ont pas toujours une sémantique formellement définie, bien que les experts ayant défini ces DSL puissent décrire l'évolution de tels systèmes. La description de leur sémantique, quand elle existe, est donc souvent décrite de façon informelle, en langage naturel. Dans le cas où cette dernière n'est pas définie explicitement, les experts du DSL doivent alors définir, au moins textuellement, la sémantique de leur DSL.

Nous considérons maintenant brièvement les combinaisons possibles de ces deux caractéristiques de sémantiques d'un DSL afin d'identifier les étapes clefs dans la définition d'une sémantique par traduction correcte. Quelle que soit la précision de la sémantique initiale, il est nécessaire qu'un expert du domaine identifie les états équivalents par rapport aux propriétés souhaitées sur le modèle. Chacune de ces classes d'équivalence est alors caractérisée par un prédicat d'état (phase d'abstraction par prédicats). Un événement ou changement d'état dans le modèle sera dit observable si les états avant et après ne sont pas équivalents. Dans la suite, la fonction Π décrit l'application de la transformation à un état d'un modèle donné. L'image d'un tel état par Π est donc ici un réseau de Petri avec un marquage particulier.

3.2. *Traduction d'une sémantique opérationelle : preuve de bisimulation*

Dans le cas où la sémantique est décrite formellement et de façon opératoire, il faut s'assurer que la sémantique d'origine et celle obtenue par traduction décrivent bien le même comportement. Il faut donc exhiber une preuve de bisimulation entre les deux sémantiques. Il s'agit d'une preuve par induction sur la syntaxe abstraite dans laquelle on montre que toute transition dans la première sémantique correspond à une transition dans la seconde, et vice-versa. Dans le cas où l'une des deux sémantiques comporte plus d'états que l'autre, il s'agit d'une bisimulation faible. Une telle preuve garantit que les événements observables des deux sémantiques sont identiques et donc que les analyses effectuées par la suite dans le modèle cible sont bien pertinentes au niveau du modèle source. La bisimulation consiste à démontrer qu'un modèle et son image par la fonction de traduction Π décrite au-dessus ont des événements observables identiques à chaque instant. On note $s_1 \Rightarrow s_2$ si s_2 est l'image de s_1 par une suite de changements d'états dont un seul est observable. Le théorème de bisimulation s'énonce comme suit : Soient X, X' et Y des états du modèle, alors $X \Rightarrow X'$ implique $\Pi(X) \Rightarrow \Pi(X')$. Et réciproquement, si $\Pi(X) \Rightarrow Y$ alors il existe X' tel que $\Pi(X') = Y$ et $X \Rightarrow X'$.

3.3. *Traduction d'une sémantique axiomatique : expression de la consistance*

Dans les autres cas, la sémantique initiale n'est pas assez précise et contrairement au cas précédent, il ne peut y avoir adéquation entre celle-ci et la sémantique obtenue par traduction. Cependant, on doit tout de même s'assurer *a minima* que le modèle cible vérifie les propriétés exprimées dans la sémantique initiale. Une sémantique axiomatique standard comporte des invariants, des préconditions et des postconditions qui doivent s'exprimer en fonction des prédicats d'états intéressants définis par l'expert. Sur un plan théorique, il n'y a plus bisimulation mais simplement simulation du modèle cible par le modèle source. Concrètement néanmoins, la sémantique axiomatique n'étant pas opératoire, il va alors s'agir d'exprimer celle-ci en tant qu'ensemble de propriétés des états observables du modèle cible, qui devront être vérifiées. On ne pourra traduire que des types de propriétés supportées dans l'espace technologique cible, ici des propriétés comportementales des réseaux de Petri. La possibilité de traduction des propriétés de la sémantique axiomatique dont dépend la correction de la sémantique par traduction de modèles est donc fortement dépendante du méta-modèle cible. Cependant, dans tous les cas, il faut pouvoir traduire des prédicats d'états (invariants par exemple), c'est-à-dire généraliser la fonction de traduction de modèles Π pour qu'elle permette de traduire non pas chaque état du modèle source individuellement, mais bien chaque prédicat d'états (i.e. un ensemble d'états) défini par l'expert comme intéressant.

4. Une approche outillée pour la validation de modèles

Dans nos expérimentations, nous avons travaillé avec un langage simplifié de description de procédé, SimplePDL. Nous avons d'une part défini une sémantique opérationnelle (expérimentée avec Kermeta et ATL). Nous avons, d'autre part, formalisé une transformation de SimplePDL vers les réseaux de Petri, définissant ainsi une sémantique de traduction. Notre objectif était alors de pouvoir traduire une propriété de SimplePDL en propriété LTL sur les réseaux de Petri pour la vérifier ensuite en utilisant le model-checker de Tina (Berthomieu *et al.,* 2004). Les propriétés vérifiées peuvent être de deux natures : existentielles ou universelles. Dans le premier cas, la propriété doit être vérifiée dans toute exécution. Si ce n'est pas le cas, l'outil fournit une trace contre-exemple de la propriété. Le second cas permet de vérifier qu'une exécution satisfait la propriété, par exemple des contraintes en temps ou en ressources. Si une telle exécution existe, une trace est générée par l'outil.

Partant de ces expérimentations, nous avons établi une preuve de bissimulation entre les deux sémantiques qui garantit que les conclusions obtenues sur les réseaux de Petri sont équivalentes à celles que l'on pourrait avoir sur le modèle SimplePDL : une propriété vérifiée par Tina est donc une propriété vérifiée sur SimplePDL.

Il est maintenant indispensable de poursuivre ces travaux dans le but d'interpréter automatiquement les résultats obtenus sur les réseaux de Petri en terme des concepts de SimplePDL. Ceci permettrait en plus de connecter les deux approches : une trace obtenue par vérification pourrait être traduite en trace du modèle initial (en Sim-

plePDL) et exploitée par le simulateur. Les traces correspondant à des contre-exemples permettent alors de trouver les erreurs du modèle source tandis que les traces d'une propriété existentielle peuvent servir pour simuler une exécution possible.

5. Conclusion

Fort de nos expériences sur l'expression de la sémantique d'un DSL, nous proposons une manière de coupler une sémantique opérationnelle et une sémantique de traduction. Pour compléter cette approche, il nous manque une étape essentielle qui consiste à pouvoir remonter les résultats obtenus avec les outils de vérification vers le DSL source. Si cette remontée peut être réalisée de manière ad'hoc dans le cas d'un DSL spécifique, il nous semble important de trouver un moyen de l'exprimer à un niveau d'abstraction suffisant pour pouvoir l'appliquer plus facilement à de nouveaux DSL. Sur ce point, nous envisageons une approche dirigée par les propriétés que l'utilisateur du DSL source souhaite étudier. Elles permettraient d'abord d'identifier les éléments du DSL source qui ne sont pas explicites (état global), aiderait la définition de la transformation vers le DSL exécutable et faciliterait la remontée d'informations. Une approche complémentaire pour exprimer une traduction bidirectionnelle à un niveau d'abstraction suffisant pourrait être l'expression d'un modèle de correspondance. Un formalisme pourrait alors être proposé à l'utilisateur du DSL pour exprimer ses propriétés, les outils de vérification étant alors complètement cachés.

6. Bibliographie

Berthomieu B., Ribet P.-O., Vernadat F., « The tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets », *Int. Journal of Production Research*, vol. 42, n° 14, p. 2741-2756, 2004.

Clark T., Evans A., Sammut P., Willans J., « Applied Metamodelling - A Foundation for Language Driven Development », 2004, version 0.1.

Combemale B., Crégut X., Berthomieu B., Vernadat F., « SimplePDL2Tina : Mise en oeuvre d'une Validation de Modèles de Processus », *in* , H. Sciences/Lavoisier (ed.), *3ieme journées sur l'Ingénierie Dirigée par les Modeles (IDM)*, Toulouse, France, 2007.

Combemale B., Rougemaille S., Crégut X., Migeon F., Pantel M., Maurel C., « Sémantique dans la méta-modélisation », *in* , H. Sciences/Lavoisier (ed.), *2ieme journées sur l'Ingénierie Dirigée par les Modeles (IDM)*, Lille, France, 2006.

Favre J.-M., Estublier J., Blay M., *L'Ingénierie Dirigée par les Modèles : au-delà du MDA*, Informatique et Systèmes d'Information, Hermes Science, 2006. 2-7462-1213-7.

Jouault F., Kurtev I., « Transforming Models with ATL », *Proceedings of the Model Transformations in Practice Workshop at MoDELS*, LNCS, Springer, Jamaica, 2005.

Muller P.-A., Fleurey F., Jézéquel J.-M., « Weaving Executability into Object-Oriented Meta-Languages », *in* , S. K. L. Briand (ed.), *MoDELS*, LNCS, Springer, Jamaica, 2005.

Taentzer G., « AGG : A Graph Transformation Environment for Modeling and Validation of Software », *in* , Springer-Verlag (ed.), *AGTIVE*, vol. 3062 of *LNCS*, p. 446-453, 2003.

Winskel G., *The formal semantics of programming languages : an introduction*, MIT Press, Cambridge, MA, USA, 1993.