Towards Automatic Aspect Introduction and Analysis Using Temporal Theories in Hets

Nour Hossain[1] and Wolfram Kahl[2]

[1] McMaster University, Hamilton, Ontario, Canada, hossaimn@mcmaster.ca
[2] McMaster University, Hamilton, Ontario, Canada, kahl@mcmaster.ca

1 Motivation

When Fiadeiro and Maibaum (1992) introduced their "temporal theories as modularisation units for concurrent system specification", they emphasised that they were "only" introducing a logic, but not a specification language. We now report on-going work to provide tool support for a Casl-based specification language using this logic in Hets (Mossakowski et al., 2007). Our immediate goal is to provide support for performing and analysing aspect introduction at the system architecture level, where system architectures are diagrams in the category of these temporal theories, and aspect introduction is performed via a special kind of diagram transformation sketched in (Hossain et al., 2017).

2 Temporal Theories for Concurrent System Specification

A temporal theory following Fiadeiro and Maibaum (1992) is formulated over a signature $\theta = (\Sigma, A, \Gamma)$, where Σ is a conventional algebraic signature, A is a separate finite set of *attribute symbols* that syntactically behave like the function/operator symbols of Σ, and Γ is a separate set of *action symbols* that syntactically behave like predicate symbols. While the interpretation of Σ is a conventional algebra, the semantics of attribute and action symbols is parameterised over time (encoded as natural numbers).

The term language is additionally enriched with a "next" operator, such that $\mathbf{X}\,t$ refers to "the value of t in the next instant". The formula language is enriched with conventional LTL operators and the new atomic formula **BEG** that holds only at time point 0. A specification (θ, Φ), where Φ is a set of θ-formulae, is called an *object description* by Fiadeiro and Maibaum (1992).

Among the θ-interpretation structures for a signature $\theta = (\Sigma, A, \Gamma)$, those satisfying the *locality axiom for* θ (where x_g and x_a represent tuples of variables of the argument sorts of g, respectively a),

$$\left(\left(\bigvee_{g\in\Gamma} (\exists\, x_g \bullet g(x_g))\right) \vee \left(\bigwedge_{a\in A} (\forall\, x_a \bullet \mathbf{X}\, a(x_a) = a(x_a))\right)\right) \ ,$$

are called *θ-loci* — they represent θ-behaviours that are "disciplined" in the sense that values of attributes in A only change at times where actions in Γ are taking place. A *model* of a specification (θ, Φ) is a θ-locus in which all axioms from Φ are *true*, that is, satisfied by every variable assignment.

A specification homomorphism from (θ_1, Φ_1) to (θ_2, Φ_2) is a signature homomorphism σ such that not only all σ-translations of axioms in Φ_1 are valid in (θ_2, Φ_2), but also the σ-translation of the locality axiom for θ_1. This condition ensures that embedding an "object" into a larger object remains modular, that is, that the control of an object over "its" attributes is not circumvented by the embedding: Locality of actions has to be preserved.

The resulting category has finite colimits; in the construction of colimits, the translation of the locality axioms of the source specifications need to be added to the translations of their axioms.

3 Adding the Logic of Temporal Theories to Hets

Even though Fiadeiro and Maibaum (1992) used the setting of algebraic specifications, without predicate symbols, it appeared natural to us to integrate their logic into Hets as an extension of Casl, which allows us to use the designed-for-extension Casl infrastructure of Hets, and also re-use much of the operator and predicate symbol infrastructure for attribute and action symbols.

We use the "views" of Casl/Hets to encode morphisms, and can generate the proof obligations, including locality preservation, for checking the well-definedness of the resulting morphisms.

Fiadeiro and Maibaum (1992) intended their temporal theories to be used for specifying concurrent systems by first constructing what we call *system architectures*, namely diagrams of object descriptions, and then considering the colimit of such diagrams as the system specification.

Hets supports colimit construction of diagrams of specification — adding locality preservation axioms to colimit specifications in this logic (to make the co-cone morphisms well-defined) is straightforward.

4 Support for System Architecture Transformation

Since our goal includes system architecture transformation via pushouts of the "zigzag system architecture homomorphisms" introduced in (Hossain et al., 2017), we need to represent diagrams of system architectures, that is, diagrams of diagrams of object descriptions. We currently achieve this by collecting each system architecture into a Hets architectural specification, and extracting the system architecture homomorphisms from the architecture-crossing views. (Recent developments in Hets, such as those reported by Calegari et al. (2016) and Codescu et al. (2017), may offer some alternatives or improvements, but still do not appear to provide a direct representation of such nested diagrams.)

The goal of performing this diagram transformation in Hets is to automate generation of property translations for the purpose of analysing the effects of aspect introduction via system architecture transformations: Since aspect introduction always intentionally breaks some (undesired) properties, we have only limited automatic property preservation, and want to be able to explore preservation of "already-good" properties across aspect introduction, and addition (via preservation from the rule right-hand side) of new desired properties created by aspect introduction.

References

D. Calegari, T. Mossakowski, N. Szasz. Heterogeneous verification in the context of model driven engineering. *Science of Computer Programming*, 126:3–30, 2016. ISSN 0167-6423. doi: 10.1016/j.scico.2016.02.003. Selected Papers from the 17th Brazilian Symposium on Formal Methods (SBMF 2014).

M. Codescu, T. Mossakowski, D. Sannella, A. Tarlecki. Specification refinements: Calculi, tools, and applications. *Science of Computer Programming*, 144:1–49, 2017. ISSN 0167-6423. doi: 10.1016/j.scico.2017.04.005.

J. Fiadeiro, T. Maibaum. Temporal theories as modularisation units for concurrent system specification. *Formal Aspects of Computing*, 4(3):239–272, 1992. ISSN 1433-299X. doi: 10.1007/BF01212304. URL `https://doi.org/10.1007/BF01212304`.

M. N. Hossain, W. Kahl, T. Maibaum. A graph transformation approach to introducing aspects into software architectures. In L. Burgueño et al. (eds.), *MODELS 2017 Satellite Events*, CEUR Workshp Proceedings, Vol-2019, pp. 54–63. CEUR, 2017. URL `http://ceur-ws.org/Vol-2019/`.

T. Mossakowski, C. Maeder, K. Lüttich. The heterogeneous tool set. In B. Beckert (ed.), *VERIFY '07, 4th International Verification Workshop*, CEUR-WS, vol. 259, pp. 11.1–11.17, 2007. URL `http://ceur-ws.org/Vol-259/`.