3.3 WG4: Multiple Interacting Bidirectional Transformations

Holger Giese (Hasso-Plattner-Institut, DE), and Gabor Karsai (Vanderbilt University, US), and Vadim Zaytsev (Raincode Labs, BE)

Most of the work of the working group was shaped, motivated and conceptually linked to two case studies presented earlier during the week of the seminar: the cyber-physical system case study by Holger Giese (cf. [2]) and the compiler case study by Vadim Zaytsev (§4.2, also cf. [3]).

It seemed to us during the numerous discussions that syntactic consistency (achievable with bx) can be viewed as a precondition for enforcing some more global constraints (name uniqueness, deadlock freedom, simulation, model checking, etc.) with multx. This idea could potentially serve as a framework to combine bx and multx into one network. Whether this or another composition framework should be used in the future, it should be related to the scheme proposed earlier in 2018 by Diskin, König and Lawford [1].

3.3.1 Taxonomy

During one of the sessions the working group has sketched a taxonomy that can be, with sufficient domain research and linking to existing literature, worked on in the future and expanded into a real taxonomy of multidirectional transformations.

- Arity (of multx)
  - 1 (internal consistency)
  - 2 (bx)
  - > 2 (“true” multx)
- Model relations/hierarchy
  - List or sequence
  - Tree
  - DAG
  - Graph
  - Hypergraph
- Consistency
  - Representation
  - Structural consistency (“syntax”)
  - Behavioural consistency (“dynamic semantics”)
  - Static semantics
- Versioning
- Change action
  - Central
  - Distributed
- Megamodel of multx
- Authority
- Conflict resolution
- Policy or strategy
  - Lossy
  - Preserving
- Precondition

- bx–multx
- multx–bx

3.3.2 Case Study

In an attempt to come up with a simpler case study that would still be capable of serving as an example to demonstrate, we came up with the following. (NB: it was technically impossible to design a truly simple scenario because truly simple scenarios are handled by bx and other simpler frameworks and do not require multx).

Our domain is that of vending machines. The entity we would like to model, is of a machine serving coffee and tea: when enough coins are inserted, and a choice is made by pressing the right button, the machine produces the desired beverage to the best of its ability. Consider three models: L, P and S.

L is a labelled transition system. Its metamodel concerns states and transitions between them, with labels attached to both. L is good at modelling such aspects of the system as various kinds of buttons that the machine may have, as well as consequences of activating them. We also assume the labels have a way of representing sending and receiving messages if buttons are to be understood as channels.

P is a Petri net. Its metamodel concerns places, transitions, arcs between them and tokens that show which transitions can be fired according to the number of incoming arcs matched against the number of available tokens. P is good at modelling counting of all kinds, such as the number of inserted coins or the number of servings of coffee that can be made from available coffee beans before the machine needs to be refilled.

S is a sequence diagram. Its metamodel concerns lifelines, agents, messages and their relative timings. S is good at modelling sequences of events without knowing the internal workings of involved agents. It can represent either valid scenarios of combining L and P, or invalid ones, or even constraints such as "there should always be a beverage served after requesting it". In general, $L \parallel P \models S$.

Tasks that are naturally implementable with bx, are of syntactic nature: for example, if one is to edit L to add new actions, such changes need to be propagated to P. Such a bx will contain, among other things, the alignment in the sense of matching elements of L to elements of P, name-based or otherwise.

Tasks that cannot be handled naturally and gracefully by bx, are global and behavioural: for example, simulating execution of L in parallel with P to see if they are capable of producing the sequence of events specified by S.

References

1 Zinovy Diskin, Harald König, Mark Lawford. *Multiple Model Synchronization with Multiary Delta Lenses*, Proceedings of the 21st International Conference on Fundamental Approaches to Software Engineering (FASE), LNCS 10802, pp. 21–37, https://doi.org/10.1007/978-3-319-89363-1_2, Springer, 2018.

2 Holger Giese, Bernhard Rumpe, Bernhard Schätz, Janos Sztipanovits. *Science and Engineering of Cyber-Physical Systems*, Dagstuhl Seminar 11441. Dagstuhl Reports 1(11), https://doi.org/10.4230/DagRep.1.11.1, 2011.

3 Vadim Zaytsev. *An Industrial Case Study in Compiler Testing.* Proceedings of the 11th International Conference on Software Language Engineering (SLE), pp. 97–102, ACM, https://doi.org/10.1145/3276604.3276619, 2018.