Short Term Scientific Report COST IC1404:
"Towards a Ultimate Formally Verified Master Algorithm"

Julien Deantoni

November 28, 2018

1. Working Group: Co-Simulation group (WP2)
2. Original title: Verification of co-simulation synchronization protocols
3. Beneficiary: Julien Deantoni
4. Host: Hans Vangheluwe
5. Period: from 28/10/2018 to 03/11/2018
6. Reference code: 42170

1 Introduction

I realized my STSM in the University of Antwerp, together with Cláudio Gomes and Hans Vangheluwe, in the context of the MPM4CPS COST Action.

The subject was contextualized by co-simulation. It is a technique that has gained attention in the recent years, because it allows multiple simulation tools to interact, enabling analyses that are otherwise costly to perform. The technique has seen rapid adoption because the tools can provide an API that allows signals to be exchanged in a black box manner. The fact that there is little information about what models are actually being coupled means that ensuring trustworthy co-simulation results is a non-trivial challenge.

During this STSM, we initially proposed to apply formal methods techniques to increase the level of confidence in co-simulation results. In concrete, we intent to work on three co-related aspects: (1) modeling the expected interaction between heterogeneous models and their actual interaction as realized by standard co-simulation algorithms, (2) formalizing correctness properties about the results of the co-simulation, and (3) applying model checking techniques to verify those properties (possibly based on a benchmark created in a previous STSM to Aarhus university).

One important aspect to enable previous points consists in characterizing the appropriate information about the models being coupled, and understanding how this information affects the satisfiability of the properties. Also, properties themselves must be defined.

We spent the STSM week having discussion on this last point. We started by trying to understand what kinds of properties must be verified by the Master Algorithm; and what pieces of information are needed to check these properties. After the week of reasoning, we end up with a strong claim that is developed in the next sections: "a co-simulation is functionally correct if no simulation time delays are introduced by the co-simulation mechanisms". While the claim may be too strong to be totally true, we believe it helps to understand most of the actual problems of co-simulation. This reports is split in two main sub-sections to highlight our results. First, we elaborated defined and explained what a co-simulation delay is, and then, based on state of the art approaches, we sketched a framework that exhibits information from which the delay property can be managed correctly. The final goal that need to be handled in further collaboration is the definition of a ultimate[1] master algorithm based on such information.

2 Advanced Realized During the STSM

2.1 Context

Nowadays, a co-simulation is a key enabler to system engineering, where models written in different languages are bundled into executable entities, connected together and coordinated by a Master Algorithm. The MA is in charge of transferring data from one entity to another and simulating each entity so that the collaborative simulation exhibits the behavior of the system. More precisely,

1. The input of a co-simulation (*co-simulation setup*) is a topology of connected entities, *i.e.*,, a directed graph of entities.

$$\text{CSS} \triangleq < \mathcal{E}nt, \mathcal{C}on > \tag{1}$$

2. Internally, an *entity* is a made up with a model and a solver. The solver implements the operational semantics of the language the model conforms to. The model and the solver inside an entity are not directly accessible and only some information on them are provided in an entity *interface*. The entity interface is a set of named *inputs* and *outputs* together with their properties in terms of *type* (*e.g.*,Boolean or Float). In order to be coordinated by the MA, each entity exposes an API. Finally, an entity exposes its internal time, *i.e.*,the simulation time inside the entity. The representation of the virtual simulation time inside all entities is usually a float [TCV+16, WME+13, CLB+16, Mod14].

$$e \in \mathcal{E}nt \triangleq < \mathcal{I}n, \mathcal{O}ut, \text{API}, internalTime \in \mathcal{R} > \tag{2}$$

$$i \in \mathcal{I}n \text{ or } o \in \mathcal{O}ut \triangleq < type > \tag{3}$$

$$type \triangleq Float|Integer|Boolean \tag{4}$$

[1] ultimate according to our problem statement, *i.e.*,the non introduction of co-simulation delay

3. An edge in the graph represent a *connector* between an input and an output:

$$con \in \mathcal{C}on \triangleq < source, target > \text{, where } source \in \mathcal{I}n \wedge target \in \mathcal{O}ut \quad (5)$$

4. By using its API, it is possible to ask to an entity to do a *simulation step* of a specific duration greater to 0,0. The entity can reject the step if, for any reason the step is too big for it (*e.g.*,a to big gap between two consecutive values of an input or a discontinuity in one of its output). It is also possible to *get* and *set* respectively the outputs and inputs of an entity, typically between two simulation steps to transfer data from one entity to another. Finally, it is optionally possible to get and set the state of the entity so that rollback to a specific, previously saved, point in time is possible[2].

$$\begin{aligned} \text{API} \triangleq < & \\ & < state, currentTime \\ & > doStep(\Delta t, stateToRestore) \text{ with } currentTime, \Delta t \\ & \in \mathcal{R}, \delta t \\ & > 0; getOutputs; setInputs \\ & > \end{aligned} \quad (6)$$

5. we consider that the CSS is correctly type, *i.e.*,$\forall c \in \mathcal{C}on, (c.tail.type = c.head.type) \wedge (c.tail.nature = c.head.nature)$ of a specific type are connected to an input of a the same type and nature.

2.2 SoTA and Problems

Various problems have been identified in the state of the art. For instance, [CLT+16, BGL+15] point the necessity to add the notion of *Event* to better deal with discrete models as well as the necessity to allow the duration of a step to be equal to 0 in order to deal with "instantaneous" reactions.

To allow the integration of discrete cyber models (*e.g.*,entities encapsulating a set of periodic tasks), [CDDS16, LDP+18, TCV+16] proposed to extend the co-simulation interface in order to allow event driven communication with them, avoiding the introduction of delays due to the co-simulation. [CDDS16] also proposed to saved the internal trace of outputs computed during a co-simulation step.

Most of these works recommend the use of super-dense time [BBG+13, CLB+17, TBS14, LSV98] to characterize instantaneous evolution of an entity.

Also, [MGVB16] proposed to provide some semantic adaptation so that the entities that are not providing the expected capabilities (*e.g.*,input approximation) can be used safely in a co-simulation.

[2] the actual APIis of course bigger but activities like loading, ending, or checking are not relevant here

Based on these extensions/problem identifications, the previously cited approaches proposed dedicated Master Algorithms, were the identified related problems were solved. However, these approaches were usually implicitly considering a topology of components with specific capabilities. For instance, to cite one, [CLB$^+$16] proposed to solve the step revision problem but his approach consider that the entities to be simulated are Mealy machines, with rollback capabilities and the possibility to sort the connectors to propagate the outputs of an entity to the input of another one. They also considered only topology of entities that does not contain an algebraical loop. However, their MA did not assume the possibility to have a Run To Completion between various entities that require the have a fix point instantaneous iteration.

Of course this approach like the other ones are nevertheless interesting and increased the level of knowledge on how to coordinate different entities under simulation. However, we believe that it is time now to put all this knowledge together in a (extensible) framework for co-simulation where all the already proposed extensions are integrated and more importantly where the capabilities of each entity is made explicit so that the MA can figure out the strategy and the semantic adaptation to adopt. In other terms, exposing the capabilities of an entity is used to attribute the responsibilities either to the entity when capable or the to the MA otherwise.

2.3 Proposition

1. As a starting point, we the definition provided in section 2.1 and extended by the State of the Art propositions as follow

   and *nature* (*e.g.*,piecewise constant or continuous).

$$override \quad i \in \mathcal{I}n \text{ or } o \in \mathcal{O}ut \triangleq < type, nature > \tag{7}$$

$$override \quad type \triangleq Float|Integer|Boolean|Event \tag{8}$$

$$nature \triangleq spurious|Piecewise - Constant|Continuous|Piecewise - Continuous|Constant \tag{9}$$

$$\begin{aligned} override \quad \text{API} \triangleq & < \\ & < state, currentTime \\ & > doStep(\Delta t, stateToRestore) \text{ with } currentTime \\ & \in (\mathcal{R}, n \\ & \in \mathcal{N}), \Delta t \\ & \in \mathcal{R}, \delta t \\ & \geq 0; getOutputs; setInputs \\ & > \end{aligned} \tag{10}$$

A major problem, even with state of the art approaches is that we do not have enough information about each entity to setup a correct MA. In order to setup a correct MA, we need to exhibit the capabilities of the entities and the usage of their inputs/outputs. In most of the cases, an existing FMU does not have all the required capabilities but the capability ensuring a correct co-simulation can be implemented in the MA (with a lake of performance and sometime of precision). Before to elaborate on the description of such extensions, we define what a correct co-simulation is.

2.3.1 Correctness of a co-simulation

This definition is important since we consider a co-simulation is correct if it does not introduce any behavior that may modify the behavior of the system in comparison to a white box simulation of the entire system. Consequently, it does not mean that the system behaves as expected or as the physical laws tell use, it only means that the co-simulation is not responsible for the problem observed. Our claim is that, if a MA do its minimal job like successfully done in state of the art approaches (*e.g.*,transferring data from output to input, simulating FMUs), the only way to introduce a problem "imputable" to the co-simulation is to add a delay in the propagation of a significant change in an output.

A co-simulation delay is a delay introduced by the fact we are using co-simulation. Let consider Picture 1, which represent two entities in a partial co-simulation setup. During the simulation ab entity, it may read inputs and provide valuation of outputs. Let us denote $@P(o1, n)$ the production time of the n^{th} valuation of $o1$. Let also denote $@R(i1, n)$ the reading time of the n^{th} valuation of $i1$. At specific points in the simulation time, the Master Algorithm can get the valuation of an entity output (denoted $@G(o1, n)$). It can also set the valuation of an entity input (denoted $@S(i1, n)$). This is usually done when there exists a connector between an input and an output. Let denote $c.src$ the source (an output by construction) of a connector c and $c.dest$ the destination (an input by construction) of a connector c.

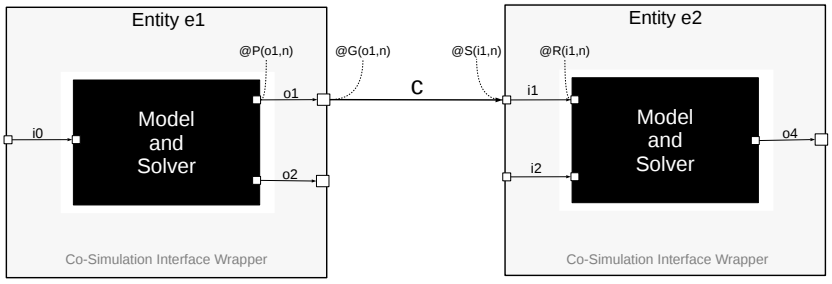

Figure 1: partial setup of a co-simulation

As examples, we provide 3 different ways for a MA to introduce a co-simulation delay, and a case where the delay is not due to co-simulation. (see

Figure 2). The first case (left of Figure 2) appears when $@S(c.src, n) - @G(c.dest, n) > 0$. In this case, the delay is due to a time consistency problem in the propagation of the valuation from one entity to another and is quite simple to avoid[3]. Another co-simulation delay appears when a significant change in an output (*e.g.*,a discontinuity) is retrieved in the MA with some delay due to the co-simulation step size (see Second example of Figure2). In this case, either the FMU should stop on the discontinuity or the MA should rollback the entity that produced the data to ask for more precision. A last example of an interesting co-simulation delay appears when $@R(c.dest, n) > @P(c.src, n) \wedge @G(c.src, n) > @R(c.dest, n)$ (see third example on Figure 2).Based on Figure 1, it means that the MA retrieved a data created in entity e1 later than its reading by entity e2. Finally, the

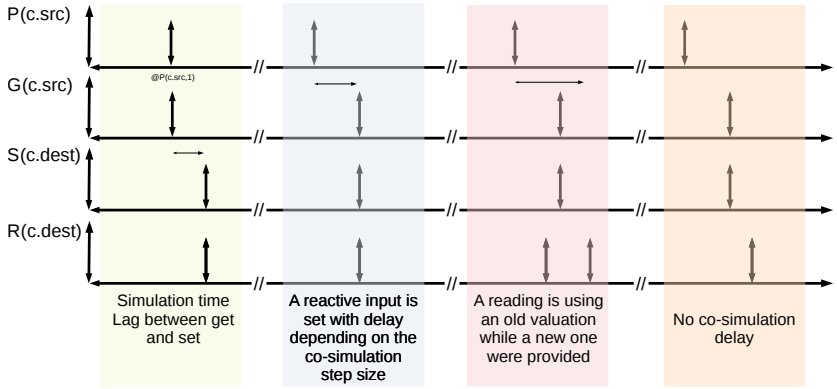

Figure 2: partial setup of a co-simulation

last example of Figure 2 is a case were there is a delay between the production of an input by an entity and its reading in another entity. However, this delay can not be inputed to the co-simulation since the delay is due to a non synchronization between the internal writing and reading of the data (*e.g.*,representing a delay due to operating system task scheduling).

In this paper we do not consider errors that are due to a bad interconnection of the entities. For instance we consider that the *type* of connected inputs/ouputs are compatible.

2.3.2 Input Usages

This section specifies what we exhibit about the model (that we expect to not reveal any important intellectual properties of the model). An *input usage* provides few but enough information of what is expected by the model.

ZCD : this usage means that the input is used to perform Zero Crossing Detection, and then that the entity is interested in a precise temporal localization

[3]note that it may also be a negative delay in such case since this is the co-siulation time that is not consistent

of the crossing.

trigger : This usage means that a new valuation of the input leads to the triggering of the internal entity behavior, usually including reading of its inputs and further production of its outputs.

sampling : This usage means that the input will be internally sampled at a given rate.

2.4 Capabilities

A *capabilities* denotes a specific behavior of an entity, associated or not to an input or output, which is related to co-simulation. For instance we define the capability for an entity to support super dense time. Based on this, we can decide if this is the MA is responsible to adapt the internal time of the entity to fit the expected super dense time or not. An example of a capability associated to an input is the capability to extrapolate the input it is associated with. Such information is of primary importance to 1) decide the data that should be provided to this input according to the internal time of the entity, and 2) know if the MA should do it or not, when required, according to the nature of the input. The reminder of this section defines the capability we identified and that will be used later to provide a MA that ensure a correct co-simulation.

2.4.1 Input Capabilities

The following capabilities are all associated to an input.

reject if unexpected change : this capability means that the entity is capable to reject an input if the difference with the previous input does not allow him to guarantee a correct behavior. This is for instance the case when an entity is using an input to do a Zero Crossing Detection (ZCD). In this case, the entity is actually trying to detect the actual time at which the input value is crossing a specific value. Then, it may found out that the difference between two successive inputs is too big to precisely enough localize the zero crossing. In this case, it will reject the step.

expose ZCD condition : While the problem is often stated as Zero Crossing Detection, it is not the case that the zero crossing are the only consideration. More often, an entity is doing a monitoring on a variable and react when this variable cross a specific value with a specific derivative sign. Exposing such condition reveals a bit of the internal model but can be used by the MA or another entity to better locate the zero crossing.

extrapolation : the capability of an entity to do extrapolation on the given input. This is of first importance to correctly feed the entity.

interpolation : the capability of an entity to do interpolation on the given input. This is of first importance to correctly feed the entity.

understand Discontinuity location : This capability means that the entity accepts to be notified that the input had a discontinuity. This is important since if this capability is associated to a ZCD, then no useless rollback is done.

2.4.2 Output Capabilities

The following capabilities are all associated to an output.

Discontinuity locator : this capability makes explicit that the entity will stop at the exact time a discontinuity is observed on the associated output.

History Provider : This capability enables the MA to retrieve not only the last computed value but all the value computed during the last step.

2.4.3 Entity Capabilities

The following capabilities are part of the entity behavior and not related to a specific input or output.

get/set state : this capability means that the entity is save and restore its entire state so that rollback, mechanism can be envisioned at the specific points in time when the MA saved them.

rollback@ : this capability means that the entity keeps a state of its state at different points in time and is able to restore it even if not saved explicitly by the MA. This make for instance sense for models written in language with omniscient debugging. Note that, in the context of co-simulation, the time to which an entity can be restored could be bounded to fit a co-simulation step.

doStep(0) : this means that the entity supports 0 communication step size.

SDT : this capability means that the entity supports super dense time, both as input for the communication step size and as output when it stops before the expected end of the step.

Mealy ? : *not clear yet if necessary*

Deployed ? : making explicit the fact that the entity introduce delay between its inputs and outputs ? *not clear yet if necessary.*

2.5 Responsabilities

Based on the different capabilities and a specific topology (see equation 1), it seems possible to elaborate a dedicated MA, which implements the appropriate mechanisms acording to the allocation of so called responsibilities. For instance, if a connector links an output with discontinuity location to an input with a zero crossing detection usage and with no rollback mechanism, the MA does not behave the same than if the input does not use the input for zero crossing detection. All this work is still under study.

3 Continuity of this Work and Further Collaboration

We already made good advances in the understanding of the properties of a correct co-simulation algorithm. We also made advances on the classification of the information (from the entities under co-simulation) that must be exhibited in to allocated the responsibilities either to a specific entity or to the MA. Allocating correctly the responsibilities may lead to a correct co-simulation.

Two main things are still to be done. The first one consists in finishing and formalizing the identification of the different properties identified in this document. This will be done in the next weeks by different telco with Claudio and Hans.

Then, based on these properties, the allocation of the responsibilities actually leads to the definition of an appropriate MA. We still have to determine the rule that lead to responsibilities allocation and the topologies for which we may or may not do a responsibility allocation that ensure the correctness of the co-simulation. This may require face to face meeting and we have to discuss how this may be done.

Finally, based on the formal definition of our concepts and beyond evidences acquired by using different examples, a proof may be realized to ensure the correctness of the approach.

it is possible to ask to an entity to do a *simulation step* of a specific duration, possibly equal to 0 to represent an arbitrary number of micro steps. If for any reason the entity reject this proposition, it should state the (super dense) time at which the step aborts as well as the reason of its rejection. The reason can be due to an unexpected input (*e.g.*,a to big gap between two consecutive values) or to an internal important event that is reported on the output (*e.g.*,a discontinuity in one of the output or the occurrence of an event). . The possibility to reject a step should be defined in what we named *capabilities* associated to an input or output. For instance if an input is used in a guard (zero crossing detection behavior), then the entity should be able to reject a step when the input varied too much from the guard between two steps: this is a *Zero Crossing Detection* capability. Also, it must reject a step if a discontinuity appeared in one of its output: this is a *Discontinuity Location* capability.

Under these assumptions, for a co-simulation to happen, a master algorithm

is in charge of simulating each entities for a specific duration and transferring data from inputs to outputs. It is also responsible for the rollback of entities when needed. Finally, it is in charge of allocating the responsibilities of each entities concerning step rejection (for instance if a *ZCD* input is connected to a *DL* output, then there is no need to roolback on discontinuity, this is useless and may lead to unexpected delays. Note that it can also assign a responsibility to himself if needed.

References

[BBG+13] David Broman, Christopher Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of FMUs for co-simulation. In *Eleventh ACM International Conference on Embedded Software*, page Article No. 2, Montreal, Quebec, Canada, 2013. IEEE Press Piscataway, NJ, USA.

[BGL+15] David Broman, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Requirements for Hybrid Cosimulation Standards. In *18th International Conference on Hybrid Systems: Computation and Control*, pages 179–188, Seattle, Washington, 2015. ACM New York, NY, USA. Series Title: HSCC '15.

[CDDS16] Stefano Centomo, Julien Deantoni, and Robert De Simone. Using SystemC Cyber Models in an FMI Co-Simulation Environment. In *19th Euromicro Conference on Digital System Design 31 August - 2 September 2016*, volume 19 of *19th Euromicro Conference on Digital System Design*, Limassol, Cyprus, August 2016.

[CLB+16] Fabio Cremona, Marten Lohstroh, David Broman, Marco Di Natale, Edward A. Lee, and Stavros Tripakis. Step Revision in Hybrid Co-simulation with FMI. In *14th ACM-IEEE International Conference on Formal Methods and Models for System Design*, Kanpur, India, November 2016. IEEE.

[CLB+17] Fabio Cremona, Marten Lohstroh, David Broman, Edward A. Lee, Michael Masin, and Stavros Tripakis. Hybrid co-simulation: It's about time. *Software & Systems Modeling*, November 2017.

[CLT+16] Fabio Cremona, Marten Lohstroh, Stavros Tripakis, Christopher Brooks, and Edward A Lee. FIDE: An FMI integrated development environment. In *31st Annual ACM Symposium on Applied Computing*, pages 1759–1766, Pisa, Italy, 2016. ACM New York, NY, USA.

[LDP+18] Giovanni Liboni, Julien Deantoni, Antonio Portaluri, Davide Quaglia, and Robert De Simone. Beyond Time-Triggered Co-simulation of Cyber-Physical Systems for Performance and Accuracy Improvements. In *10th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, Manchester, United Kingdom, January 2018.

[LSV98] Edward A Lee and Alberto Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 17(12):1217–1229, 1998.

[MGVB16] S. Mustafiz, C. Gomes, H. Vangheluwe, and B. Barroca. Modular design of hybrid languages by explicit modeling of semantic adaptation. In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*, pages 1–8, April 2016.

[Mod14] Modelisar. FMI for Model Exchange and Co-Simulation, July 2014.

[TBS14] Stavros Tripakis, David Broman, and Computer Sciences. Bridging the Semantic Gap Between Heterogeneous Modeling Formalisms and FMI. Technical report, 2014.

[TCV+16] Jean-Philippe Tavella, Mathieu Caujolle, Stephane Vialle, Cherifa Dad, Charles Tan, Gilles Plessis, Mathieu Schumann, Arnaud Cuccuru, and Sebastien Revol. Toward an accurate and fast hybrid multi-simulation with the FMI-CS standard. In *21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–5, Berlin, Germany, September 2016. IEEE.

[WME+13] E. Widl, W. Müller, A. Elsheikh, M. Hörtenhuber, and P. Palensky. The FMI++ library: A high-level utility package for FMI for model exchange. In *Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6, Berkeley, CA, USA, 2013. IEEE.