Hierarchical Continuous Time Co-simulation Optimization

Cláudio Gomes, Joachim Denil, Bart Meyers, Hans Vangheluwe

Modeling, Simulation, and Design Lab (MSDL)

Universiteit Antwerpen

Good afternoon, I'm Claudio Gomes, a PhD Student under the supervision of Prof. Hand Vangheluwe, from the University of Antwerp. I've been working in this field for two years now, but officially I got my PhD project accepted last year, so I have a bit more time to work on such a large endeavor. This work is but a small part of an ambiguous plan to develop the theory and techniques for hybrid co-simulation.

Why?

Motivation

- Simulation has helped us so far...
- ...but not to its full potential.
- Complex systems *have* to be partitioned into sub-systems, developed by specialized teams.
  - Their own M&S tools;
  - Some are external companies;
- Leading to locally (but not globally) optimal solutions:
  - Models of each partial solution cannot be integrated;
  - IP cannot be cheaply disclosed;

Simulation has brought a tremendous increase in productivity when developing systems.
However, as the complexity increases, these have to be partitioned into sub-systems, that are developed by specialized teams, or provided by external suppliers.
While this is a natural approach, it poses interesting challenges when it comes to applying modeling and simulation techniques to the coupled system as a whole:

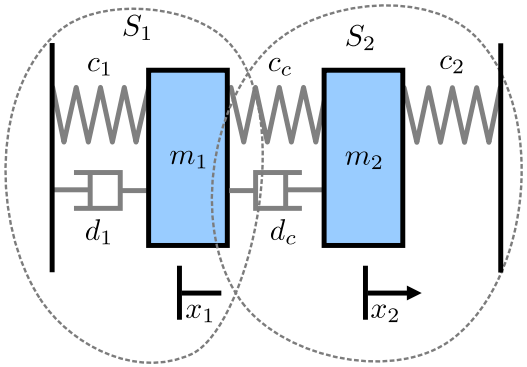
- The teams/suppliers have their own domain specific tools to develop, simulate and optimize the solutions to their sub-systems.
- The suppliers have intellectual property that cannot be disclosed.

Finally, the fact that the sub-systems are developed independently, under rigid interfaces, allows for locally optimal solutions to be found, instead of globally optimal ones.

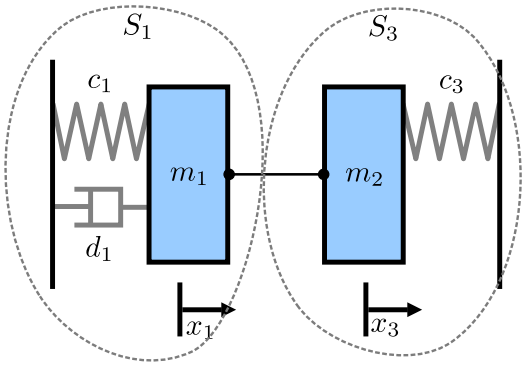
Co-simulation

- Theory and techniques to enable global simulation of a coupled system, via the composition of sub-system simulators.
- Sub-system simulators are virtual mock-ups:
  - Executable binaries;
  - Common API for communication...
  - ...but many different capabilities!

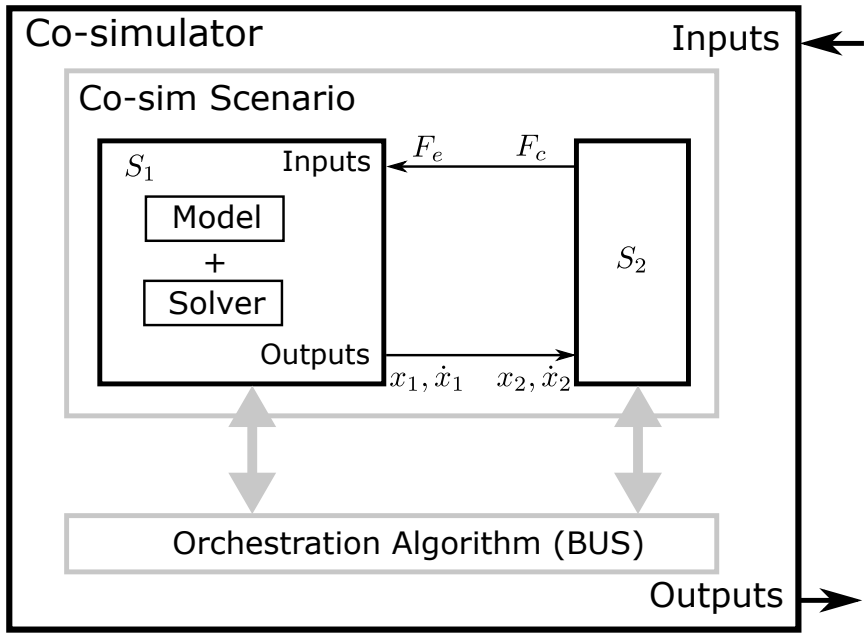
I/O Coupling:



Algebraic Coupling:



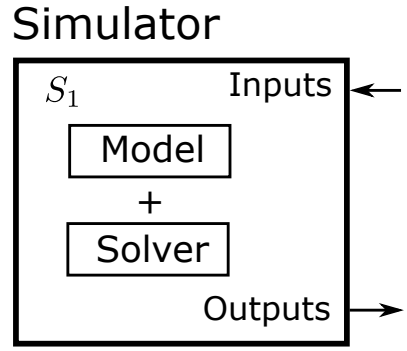
I/O Coupling Co-simulator:



Co-simulation is a pragmatic approach to solve these challenges.
It enables the simulation of the coupled system from a set of collaborating black box simulators. We're focusing on continuous time coupled systems. These are two academic examples of such systems showing the different kinds of coupling.
This first system is composed of two sub-systems, each with its own simulator.
Each simulator packs a model and a solver, and has some inputs and outputs.
For this simulator, the model is the set of differential equations for the mass-spring-damper and the solver is some numerical solver such as the Forward Euler.
Simulators have inputs and outputs, and are otherwise assumed to be independent, so an orchestration algorithm is necessary to move data around.
To properly support hierarchical co-simulation, the composition of a scenario and an orchestration algorithm has to behave as a simulator.

Simulator

- Time-stepped communication;
- Continuous-time dynamics;
- Approximated inputs;
- Physical laws;
- Instantaneous reactions;



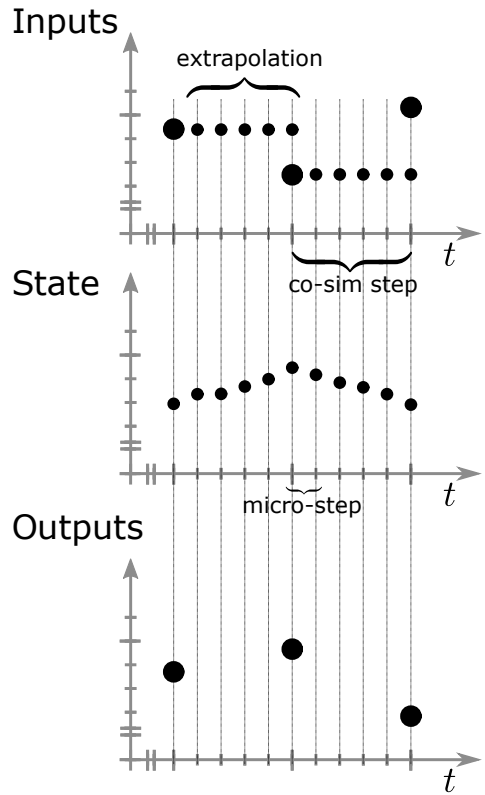
$$S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i} \rangle$$

$$\delta_i : T \times \{H\} \times X_i \times U_i \to X_i$$

$$\lambda_i : T \times X_i \times U_i \to Y_i$$

$$x_i(0) \in X_i$$

$$\phi_{U_i} : T \times U_i \times \ldots \times U_i \to U_i$$

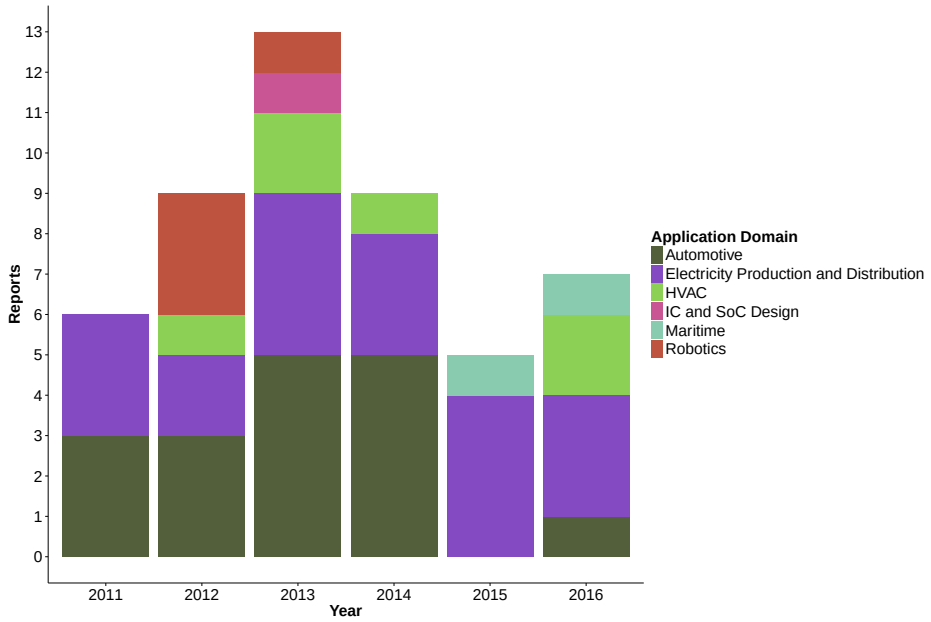


A simulator provides a simple API for setting inputs, stepping through time and getting outputs.
In the most general case, the simulator performs internal simulation steps, and extrapolates the inputs across those micro-steps.

Naturally these simulators stand for physical systems, so they have to obey to physical laws.

State of the art

- Many reported applications;
- Most involve a small number ($\approx$2) of simulators. . .
- . . . or homogeneous set of capabilities;
- Not up-to-par with CPS scale!

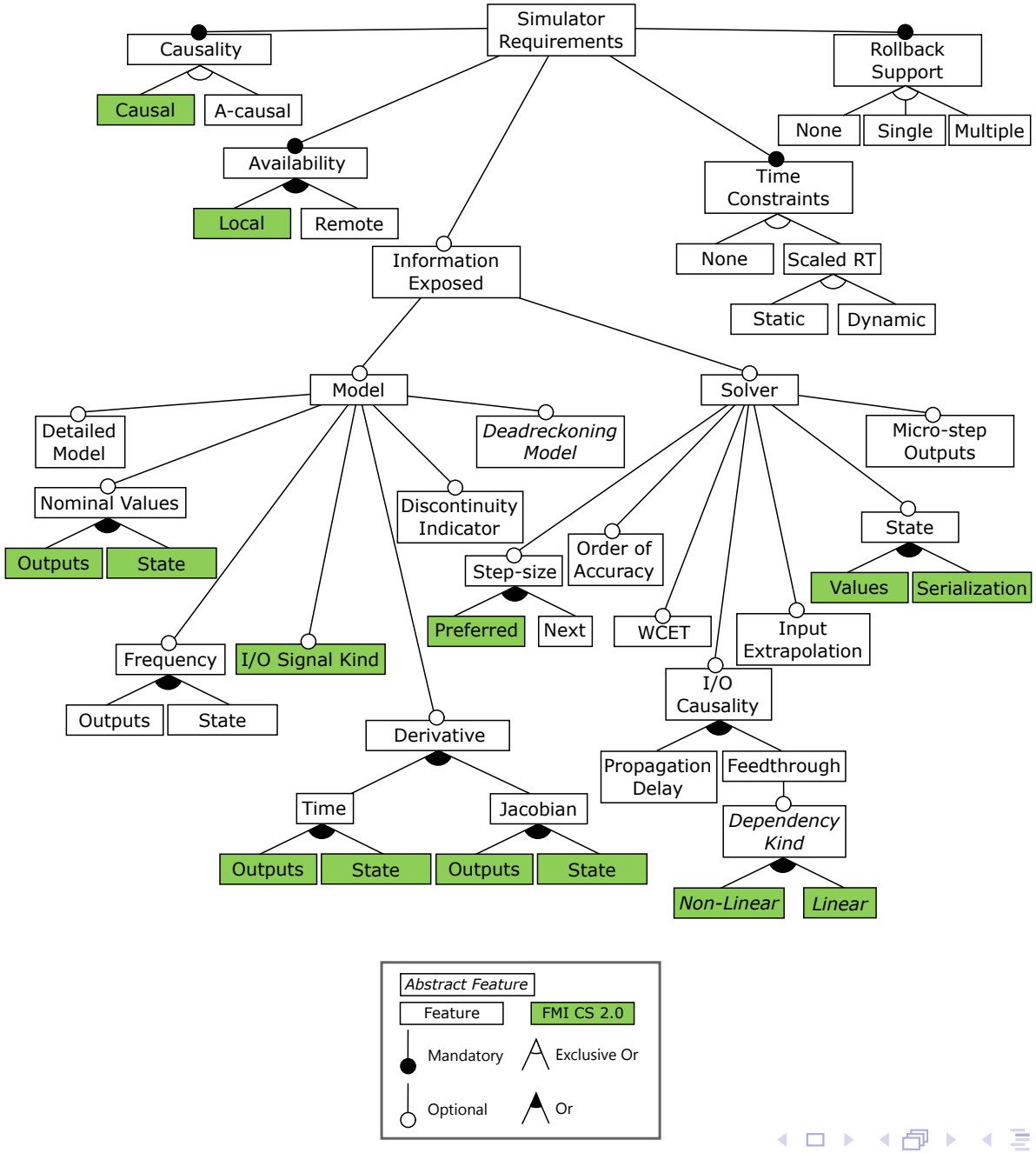


We went through a portion of the state of the art and collected the reports of co-simulation being applied in industrial case studies.
These reports have increased over the years, as you can see here. This last bar includes the last two years only.

However, most of the use cases involve either a small number of simulators, or simulators with a restricted set of capabilities, which falls short in the development of truly complex systems.

What?

Capabilities



So what makes the development of an orchestration mechanism for continuous time co-simulation so interesting? There are conflicting concerns that need to be addressed, and there is a combinatorial explosion of extra capabilities provided by simulators that need to used to address the challenges.

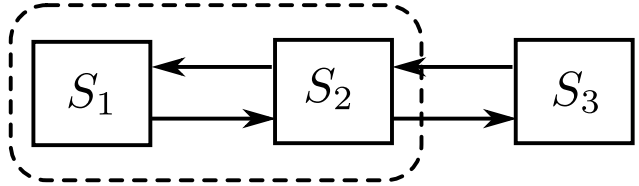
As you can see, the simulators being used in the state of the art provide many more extra features when compared to those assumed by the Functional Mockup Interface standard, highlighted in green.

Concerns

Strongly Coupled Clusters:

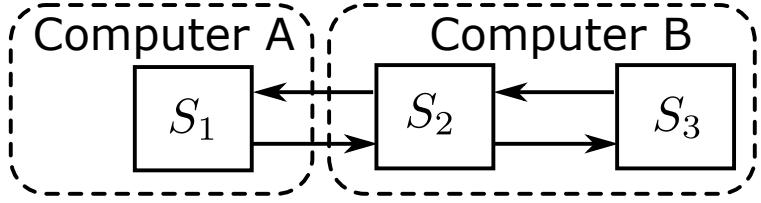
Requires: Jacobian

$$\left\{\frac{\partial F_1}{\partial Y_2}, \frac{\partial F_2}{\partial Y_1}\right\} \gg \left\{\frac{\partial F_2}{\partial Y_3}, \frac{\partial F_3}{\partial Y_2}\right\}$$

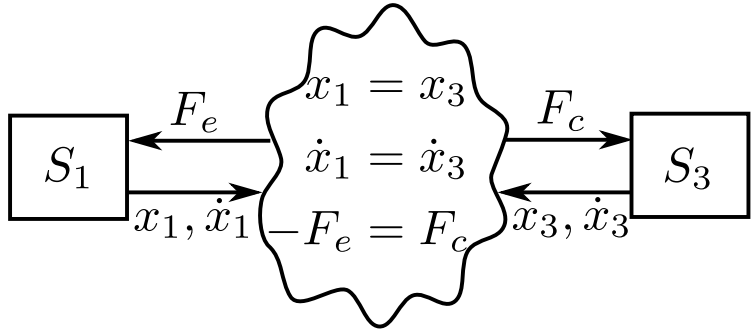


Limited Communication:

Requires: Availability

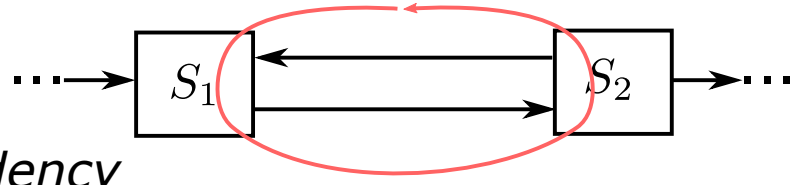


Causality Conflict:



Algebraic Loops:

Requires: I/O Dependency



These are the concerns that I was talking about. Properly addressing them increases the predictive power of the co-simulation, which is VERY important: if someone is building a new coupled system, from well known sub-systems, he/she wants the co-simulation to reflect what the coupled system will do. To recognize and solve these situations, an orchestration mechanism requires extra capabilities from the simulator.

To give a quick example, some sub-systems are more sensitive to each other than others. The more sensitive a simulator is, the more frequently it should get inputs to keep it from accumulating too large errors, so simulators should be grouped in clusters that communicate more often. To recognize and form clusters, information about the sensitivity is necessary, and that is given in some simulators in the form of a Jacobian of the system.

...

How?

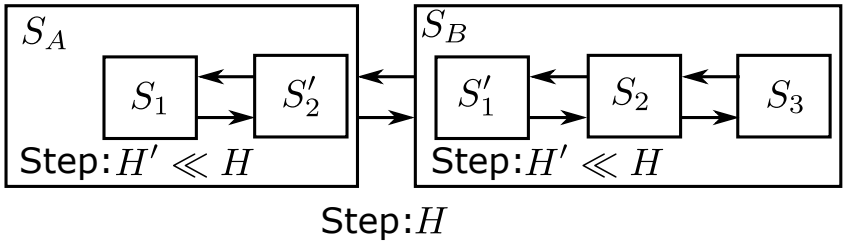
Overview

- Orchestration of co-simulation is a complex challenge:
  - Multiple concerns to address;
  - Require heterogeneous features of simulators;
- Existing orchestration mechanisms assume homogeneous set of capabilities;
- Our approach: address concerns by adding artificial simulators, and leverage existing orchestration mechanisms.

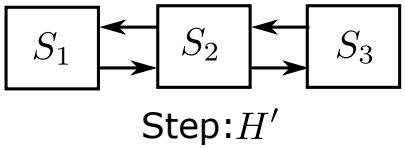
Ok so creating an orchestration algorithm is a relevant and interesting challenge because there is a variety of extra capabilities provided by simulators, and that need to be provided to solve a number of concerns in order to avoid compromising the predictive power of the co-simulation. Existing orchestration mechanisms assume a restricted set of features, which makes it difficult to recognize and address the concerns that we've described before. Our solution is based in the application of model based development to address the concerns by rewriting the scenarios, adding artificial simulators and optimizing the parameters, and then give those scenarios to existing orchestration mechanisms, so that they can be simulated.

Solutions

Limited Communication:



vs

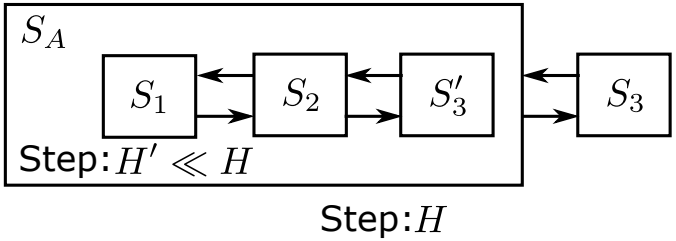


Cost: $\mathcal{O}(\frac{t_f}{H} c_{1\leftrightarrow 2})$

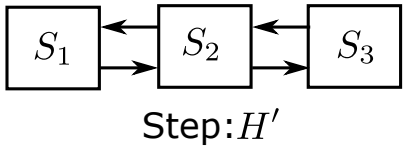
Cost: $\mathcal{O}(\frac{t_f}{H'} c_{1\leftrightarrow 2})$

Requires: Availability

Strongly Coupled Clusters:



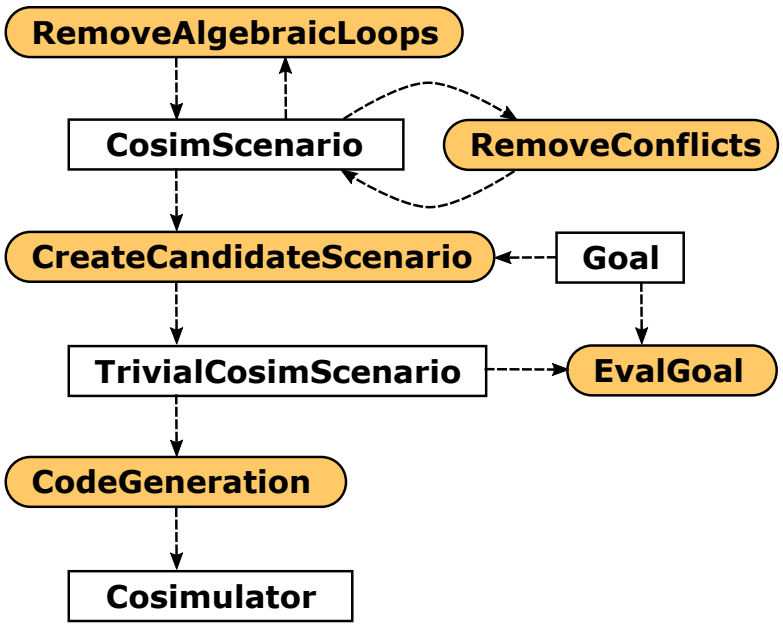
vs

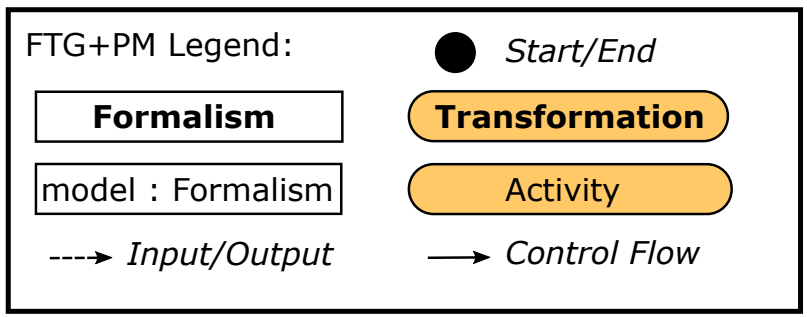


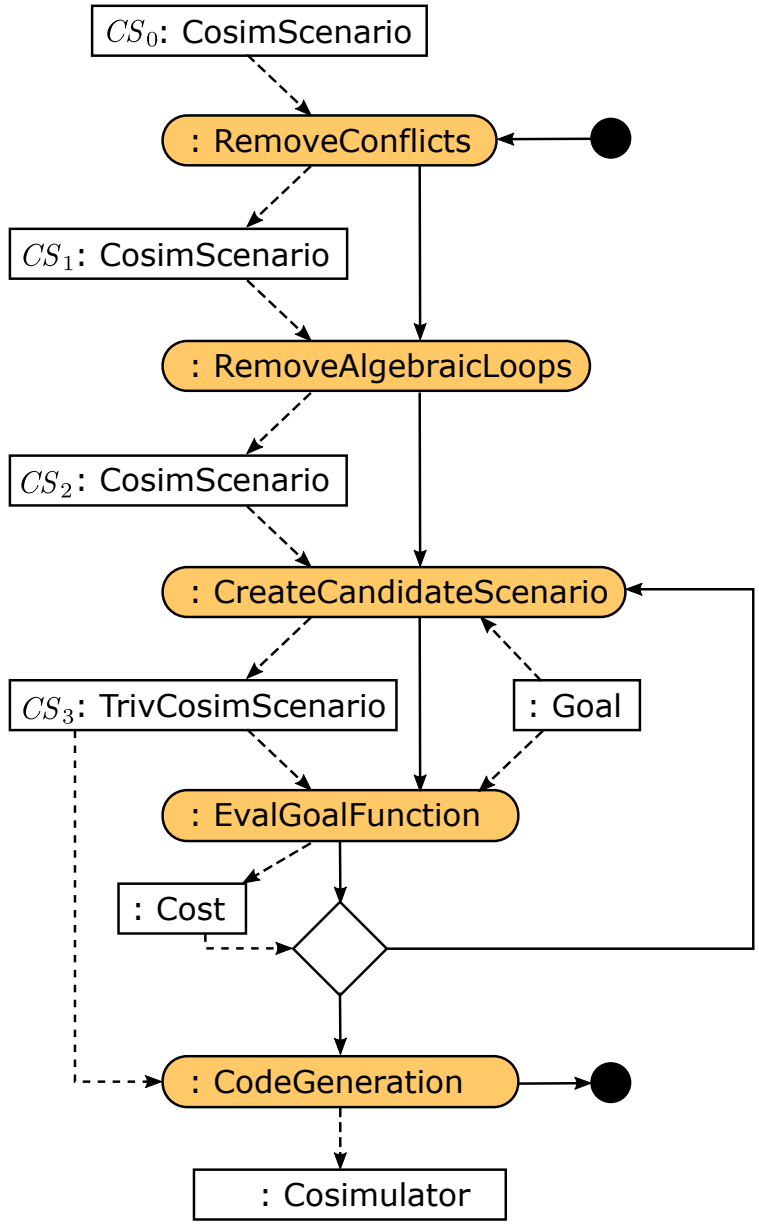
Requires: Jacobian

In the sensitive sub-systems example, we couple each cluster with a local orchestration algorithm that makes the simulators inside the cluster communicate more often. From the outside, this is a black box simulator. Since these sub-systems also need data from the third sub-system, we introduce an artificial proxy simulator, which mimics, at a higher rate, what the foreign simulator is doing. To do its jobs well, more data has to be exchanged when the cluster and the third simulator communicate. However, more data communicated less often still yields a performance benefit while keeping an approximate level of accuracy.

Big Picture


RemoveAlgebraicLoops
CosimScenario
RemoveConflicts
CreateCandidateScenario
Goal
TrivialCosimScenario
EvalGoal
CodeGeneration
Cosimulator


FTG+PM Legend:
Start/End
Formalism
Transformation
model : Formalism
Activity
Input/Output
Control Flow


CS_0: CosimScenario
: RemoveConflicts
CS_1: CosimScenario
: RemoveAlgebraicLoops
CS_2: CosimScenario
: CreateCandidateScenario
CS_3: TrivCosimScenario
: Goal
: EvalGoalFunction
: Cost
: CodeGeneration
: Cosimulator

This is a process model in the FTG+PM language. On the left you see the main languages and transformations between them. And on the right side, there is the concrete process. Overall, we establish an order in which the independent concerns are addressed, to ensure that they do not resurface while rewriting the scenarios. The conflicting concerns, such as the clustering and distribution, are solved via optimization to satisfy the goals of the user.

In the end, one co-simulation scenario is generated, to be consumed by a simple generic orchestration algorithm.

Conclusion

- Our approach, underpinned by MDD:
  - Introduce artificial simulators to solve local concerns;
  - Optimize conflicting concerns at global level;
- Correctness verified via:
  - Analytical solutions with toy examples;
  - Simulation of the coupled model;
  - High accuracy co-simulation;
- Benefits:
  - Leverage existing standards for co-simulation;
  - Systematically address concerns while reusing existing orchestration algorithms;
- Downsides:
  - Keep scenarios readable;
  - Huge search space for conflicting concerns;

The two biggest benefits of this approach is that the concerns can be dealt with, leveraging extra capabilities of simulators, independently of what is assumed by the existing orchestration mechanisms.

In the process, the scenarios become unreadable, due to the extra simulators being introduced, which do not correspond to any physical part of the system.

Also, when optimizing the co-simulation scenario for accuracy and performance, the search space is huge, and we are working to come up with heuristics to accelerate this process.

Thank you!

Bibliography

[1] C. Gomes, "Foundations for Co-simulation – IWT Proposal," University of Antwerp, Antwerp, Tech. Rep., 2015.

[2] T. Blochwitz, M. Otter, J. Åkesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel, "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models," in *Proceedings of the 9th International MODELICA Conference*. Munich, Germany: The Modelica Association, 2012, pp. 173 – 184.

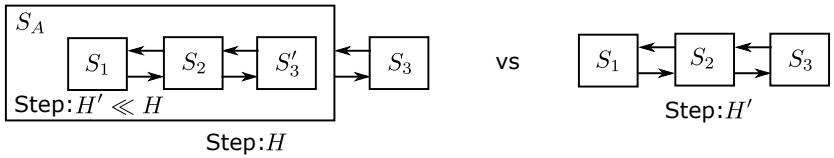
[3] H. Van der Auweraer, J. Anthonis, S. De Bruyne, and J. Leuridan, "Virtual engineering at work: the challenges for designing mechatronic products," *Engineering with Computers*, vol. 29, no. 3, pp. 389–408, 2013.

[4] B. Van Acker, J. Denil, P. D. Meulenaere, H. Vangheluwe, B. Vanacker, and P. Demeulenaere, "Generation of an Optimised Master Algorithm for FMI Co-simulation," in *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative*. Society for Computer Simulation International, 2015, pp. 946–953.

[5] K. Vanherpen, J. Denil, P. De Meulenaere, and H. Vangheluwe, "Design-Space Exploration in Model Driven Engineering: An Initial Pattern Catalogue," in *Proceedings of the First International Workshop on Combining Modelling with Search- and Example-Based Approaches (CMSEBA)*, ser. co-located with 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014). CEUR Workshop Proceedings (Vol-1340), sep 2014, pp. 42–51.

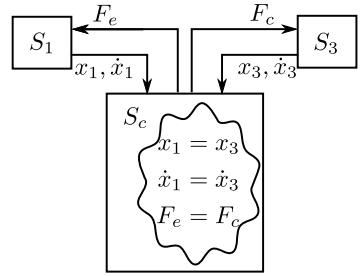
Other Solutions

Strongly Coupled Clusters:



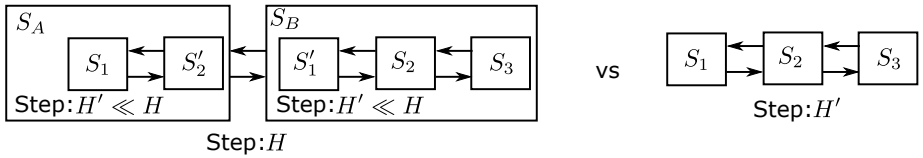
Requires: Jacobian

Causality Conflict:



Requires: Jacobian, Rollback

Limited Communication:

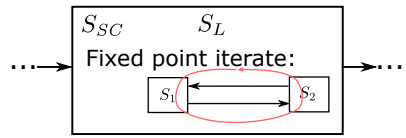


Cost: $\mathcal{O}(\frac{t_f}{H} c_{1 \leftrightarrow 2})$

Cost: $\mathcal{O}(\frac{t_f}{H'} c_{1 \leftrightarrow 2})$

Requires: Availability

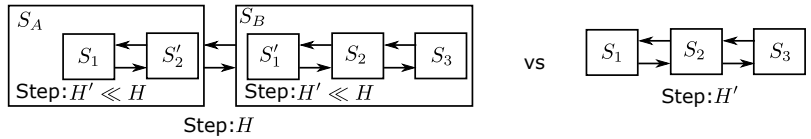
Algebraic Loops:



Requires: Rollback

Cost distribution concern

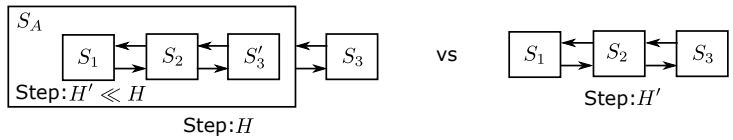
Limited Communication:



Cost: $\mathcal{O}(\frac{t_f}{H} c_{1\leftrightarrow 2})$ *Cost:* $\mathcal{O}(\frac{t_f}{H'} c_{1\leftrightarrow 2})$

Requires: Availability

Strongly Coupled Clusters:

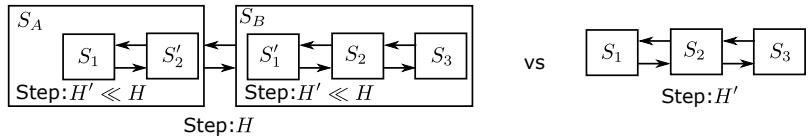


Requires: Jacobian

$$\frac{t_f}{H'}\left(\frac{H'}{h_1}c_1 + \frac{H'}{h_2}c_2 + \frac{H'}{h_3}c_3 + c_{1\leftrightarrow 2} + c_{2\leftrightarrow 3}\right) = \frac{t_f}{h_1}c_1 + \frac{t_f}{h_2}c_2 + \frac{t_f}{h_3}c_3 + \frac{t_f}{H'}c_{1\leftrightarrow 2} + \frac{t_f}{H'}c_{2\leftrightarrow 3} \approx \mathcal{O}\left(\frac{t_f}{H'}c_{1\leftrightarrow 2}\right) \quad (1)$$

Improved cost distribution concern

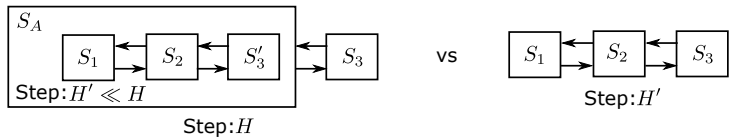
Limited Communication:



Cost: $\mathcal{O}(\frac{t_f}{H} c_{1\leftrightarrow 2})$ *Cost:* $\mathcal{O}(\frac{t_f}{H'} c_{1\leftrightarrow 2})$

Requires: Availability

Strongly Coupled Clusters:



Requires: Jacobian

$$\frac{t_f}{H}\left(\frac{H}{H'}\left(\frac{H'}{h_1}c_1 + c_2' + c_{1\leftrightarrow 2'}\right) + \frac{H}{H'}\left(c_1' + \frac{H'}{h_2}c_2 + \frac{H'}{h_3}c_3 + c_{1'\leftrightarrow 2} + c_{2\leftrightarrow 3}\right) + c_{A\leftrightarrow B}\right) =$$

$$\frac{t_f}{H'}\left(\frac{H'}{h_1}c_1 + c_2' + c_{1\leftrightarrow 2'}\right) + \frac{t_f}{H'}\left(c_1' + \frac{H'}{h_2}c_2 + \frac{H'}{h_3}c_3 + c_{1'\leftrightarrow 2} + c_{2\leftrightarrow 3}\right) + \frac{t_f}{H}c_{A\leftrightarrow B} \approx \quad (2)$$

$$\mathcal{O}\left(\frac{t_f}{H}c_{A\leftrightarrow B}\right) \approx \mathcal{O}\left(\frac{t_f}{H}c_{1\leftrightarrow 2}\right)$$