Product lines of dataflows

Extended Abstract

Michael Lienhardt
michael.lienhardt@onera.fr
ONERA
France

Maurice H. ter Beek
maurice.terbeek@isti.cnr.it
CNR–ISTI, Pisa
Italy

Ferruccio Damiani
ferruccio.damiani@unito.it
University of Turin, Turin
Italy

ABSTRACT

This one-page document summarizes a paper published in JSS [1].

CCS CONCEPTS

• **Software and its engineering → Formal methods**; **Software product lines**.

KEYWORDS

Software Product Lines, Delta-Oriented Programming, Dataflows

ACM Reference Format:
Michael Lienhardt, Maurice H. ter Beek, and Ferruccio Damiani. 2024. Product lines of dataflows: Extended Abstract. In *Proceedings of 28th ACM International Systems and Software Product Lines Conference (SPLC'24)*. ACM, New York, NY, USA, 1 page. https://doi.org/XXXXXXX.XXXXXXX

Data-centric parallel programming models like *dataflows* are well established to implement complex concurrent software. However, in case of a configurable software, the dataflow used in its computation might vary w.r.t. the selected options. One industrial tool exhibiting such a variable dataflow is *elsA*. This tool simulates the flow of fluids in a given mesh and outputs information of interest to the user (e.g., the pressure that a material pushed by the fluid must be able to sustain, or some modification of its shape that would make the fluid flow more efficiently): the core dataflow of this tool has an infinite configuration space structured in three parts.

The first part of the configuration space consists of about 2000 options that configure which fluid flow computation to perform. Indeed, fluid flow is given by Navier-Stokes equations that have no analytic solution, so hundreds of approximations of these equations have been defined, with various precision and stability characteristics: it is up to the user to decide which approximation to use.

The second part is the output information provided to the user: virtually any data could be of interest since it depends on which phenomena is studied. So the user must provide a data list to *elsA*, which in turn must compute them by extending its dataflow.

The last part of the configuration space is the shape of the input mesh itself. Meshes are structured in *zones* (modelling the domain in which the fluids flow) and *boundaries* (modelling walls of different materials, fluid injection or extraction, or even infinite domains): fluid flow simulation must be performed on every zone of the mesh and specific computations must be performed on each boundary depending on its type (e.g., the effect of a wall on a flow is different from that of a fluid injection).

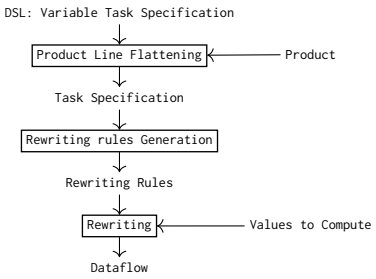

Figure 1: Dataflow generation pipeline

In this paper, we propose an approach to automatically generate dataflows given a configuration space close to that of *elsA*: instead of considering an arbitrary input mesh, we assume its variability space can be expressed with features. Our approach is structured in two main parts: first, it uses *Software Product Line* (SPL) techniques to express the variability of tasks w.r.t. the configuration space, and configures them given the options selected by the user; then, it uses *term rewriting* to assemble these configured tasks into a dataflow that computes the data requested by the user.

Figure 1 illustrates our approach. We use a *Domain Specific Language* (DSL) duly extended with concepts from *Delta-Oriented Programming* (DOP) to model the variability of the dataflow's tasks. This DSL allows us to specify which tasks, with which inputs and which outputs, are available to construct a dataflow. Then, given an input *Product* specifying the values of the different options, the *Product Line Flattening* process automatically generates the *Task Specification* corresponding to that specific product. This specification is then automatically translated into term *Rewriting Rules* by the *Rewriting rules Generation* process. Finally, given a list of *Values to Compute*, we simply apply the generated *Rewriting rules* on this data to obtain a correct *Dataflow* computing these values by using the tasks available in the specification.

In this paper, we moreover describe several analyses that check the consistency of the variable task specifications expressed in the DSL and ensure termination of the dataflow generation process (the *Rewriting* step in particular). Finally, we present an empirical evaluation of a prototype implementation designed together with the *elsA* development team. For this evaluation, we randomly chose 597 products, then generated the corresponding dataflows, and discuss the size of the generated dataflows and the execution time.

REFERENCES

[1] Michael Lienhardt, Maurice H. ter Beek, and Ferruccio Damiani. 2024. Product lines of dataflows. *Journal of Systems and Software* 210 (2024), 111928. https://doi.org/10.1016/j.jss.2023.111928

SPLC'24, September 2-6, 2024, Luxembourg

ACM ISBN 978-1-4503-XXXX-X/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX