11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
July 20–25, 2014, Barcelona, Spain

STELA: HOW A PERFORMANCE DEDICATED LANGUAGE CAN HELP TO AUTOMATE THE DEVELOPMENT OF SOLVERS AND FIELD MANIPULATORS

H. Leclerc and S. Amrouche

LMT-Cachan, 61 av. President Wilson 94235 Cachan France,
{leclerc,amrouche}lmt.ens-cachan.fr

Key words: *Programming languages, finite elements, compilation, metaprogramming, automation of computational modeling, symbolic computing.*

The C++ language is still the ultimate reference in terms of performance and flexibility. As a matter of fact, it is in the core of a vast majority of projects in our community. Among all the great features that come from the C++ language, metaprogramming[1] is a spectacular example of what can be done to gather in the same place bare metal performance and wide genericity. This feature has notably been exploited with great benefits to automate development of finite element solvers (as e.g. in [2]).

The compilation cost associated with such approaches may unfortunately become prohibitive. Furthermore, and more basically, concision and clarity of the syntax of template metaprogramming is far from what can be expected from a modern programming language.

Stela is a performance devoted general purpose programming language that was developed to adress these issues. To summarize in a few words, in order to dramatically ease and encourage the development of *active libraries* (as defined in [3]), almost every idea is stolen from C++, apart from

- the syntax (closer to coffeescript),
- the choices by default (as `template`, `auto` and `constexpr` which are activated by default),
- the compilation system (a server which stores the generated asm code separately for each function or class specialisation),

- and some pragmatic additions (as e.g. methods that can be `virtual` and `template` at the same time, tools for auto-tuning, access to expression trees, `for` loop surdefinitions, ...).

Theses tools has recently been used to develop prototype libraries for the PGD methods (as defined in [4]) where a lot of different kind of fields (including e.g. finite element fields with different axes combinations) has to communicate in pre, post and solvers phases. The basic idea is that any field in the mathematical expressions may become an unknown, leading to symbolic expressions that permit to automatically generate optimized code.

The purpose of this contribution is to show the specificities of the Stela language that have been used to design the corresponding automation tools. This approach will be compared to what can be seen with mixed python/C++ (as e.g. in [5]) or with full C++ (as mentionned before).

The shown results will be focused on compilation time, code simplicity and execution speed. Detailed explanation will permit to understand how the language works and more particularly what are the news and main differences with C++ and Python for the development of the aforementioned kinds of numerical analysis tools.

REFERENCES

[1] D. Abrahams and A. Gurtovoy. *C++ template metaprogramming: concepts, tools, and techniques from Boost and beyond. Pearson Education*, 2004.

[2] C. Prud'Homme, V. Chabannes and V. Doyeux, M. Ismail, A. Samake, G. Pena. *Feel++: A Computational Framework for Galerkin Methods and Advanced Numerical Methods. ESAIM: Proceedings*, **38**, 429–455, 2012.

[3] T. L. Veldhuizen and D. Gannon. *Active Libraries: Rethinking the roles of compilers and libraries. SIAM Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing (OO'98)*, 1998.

[4] N. Relun, D. Neron and P.A. Boucard. *A model reduction technique based on the PGD for elastic-viscoplastic computational analysis. Computational Mechanics.* Vol 51. Num 1. Pages 83-92. 2013

[5] A. Logg, K.-A. Mardal, G. N. Wells et al. *Automated Solution of Differential Equations by the Finite Element Method* . Springer. doi:10.1007/978-3-642-23099-8, 2012