RESEARCH CENTRE

Nancy - Grand Est

IN PARTNERSHIP WITH:

Université de Strasbourg

2021 ACTIVITY REPORT

Project-Team CAMUS

Compiling for Multicore Architectures

IN COLLABORATION WITH: ICube

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Architecture, Languages and Compilation

Contents

Project-Team CAMUS 1

1 Team members, visitors, external collaborators 2

2 Overall objectives 3

3 Research program 3
3.1 Static Parallelization and Optimization . . . 4
3.2 Profiling and Execution Behavior Modeling . . . 4
3.3 Dynamic Parallelization and Optimization, Virtual Machine . . . 5
3.4 Proof of Program Transformations for Multicore Programs . . . 5

4 Application domains 5

5 Highlights of the year 5
5.1 Awards . . . 5

6 New software and platforms 6
6.1 New software . . . 6
6.1.1 SPETABARU . . . 6
6.1.2 TRAHRHE . . . 6
6.1.3 TLC . . . 6
6.1.4 CFML . . . 7

7 New results 7
7.1 Improvements of the C programming language . . . 7
7.2 Ordered Read-Write Locks for applications . . . 7
7.3 Ionic Models Code Generation for Heterogeneous Architectures . . . 7
7.4 A Polyhedral Programmable Scheduler . . . 8
7.5 Automatic Task-Based Parallelization using Source to Source Transformations . . . 8
7.6 A fast vectorized sorting implementation based on the ARM scalable vector extension (SVE) 9
7.7 Shape- and scale-dependent coupling between spheroids and velocity gradients in turbulence 9
7.8 Speculative Rewriting of Recursive Programs as Loop Candidates for Efficient Parallelization and Optimization Using an Inspector-Executor Mechanism . . . 9
7.9 Algebraic Loop Tiling . . . 10
7.10 OptiTrust: Producing Trustworthy High-Performance Code via Source-to-Source Transformations . . . 10
7.11 A New Syle of Operational Semantics for Nondeterministic Languages . . . 10
7.12 Formal Verification of a Transient Data Structure . . . 11
7.13 Raster Image Processing (RIP) Optimization . . . 11

8 Partnerships and cooperations 11
8.1 European initiatives . . . 11
8.1.1 MICROCARD . . . 11
8.1.2 TEXTAROSSA . . . 12
8.2 National initiatives . . . 12
8.2.1 ANR Vocal . . . 12
8.2.2 ANR AUTOSPEC . . . 13

9 Dissemination 14
9.1 Promoting scientific activities . . . 14
9.1.1 Journal . . . 14
9.1.2 Scientific expertise . . . 14
9.1.3 Research administration . . . 14
9.2 Teaching - Supervision - Juries . . . 14

9.2.1 Teaching . . . 14
9.2.2 Supervision . . . 15
9.2.3 Juries . . . 15
9.3 Popularization . . . 16
9.3.1 Castor Informatique Contest . . . 16
9.3.2 Modern C Book . . . 16

10 Scientific production **16**
10.1 Major publications . . . 16
10.2 Publications of the year . . . 17
10.3 Other . . . 19
10.4 Cited publications . . . 19

Project-Team CAMUS

Creation of the Project-Team: 2019 March 01

Keywords

Computer sciences and digital sciences

A1.1.1. – Multicore, Manycore

A1.1.4. – High performance computing

A2.1.1. – Semantics of programming languages

A2.1.6. – Concurrent programming

A2.2.1. – Static analysis

A2.2.4. – Parallel architectures

A2.2.5. – Run-time systems

A2.2.6. – GPGPU, FPGA...

A2.2.7. – Adaptive compilation

A2.4. – Formal method for verification, reliability, certification

Other research topics and application domains

B4.5.1. – Green computing

B6.1.1. – Software engineering

B6.6. – Embedded systems

1 Team members, visitors, external collaborators

Research Scientists

- Bérenger Bramas [Inria, Researcher]
- Arthur Charguéraud [Inria, Researcher]
- Jens Gustedt [Inria, Senior Researcher, HDR]

Faculty Members

- Philippe Clauss [Team leader, Univ de Strasbourg, Professor, HDR]
- Alain Ketterlin [Univ de Strasbourg, Associate Professor]
- Vincent Loechner [Univ de Strasbourg, Associate Professor]
- Éric Violard [Univ de Strasbourg, Associate Professor, HDR]

Post-Doctoral Fellow

- Salwa Kobeissi [Univ de Strasbourg, from Sep 2021]

PhD Students

- Clement Flint [Univ de Strasbourg]
- Salwa Kobeissi [Univ de Strasbourg, until Aug 2021]
- Garip Kusoglu [Inria, from Oct 2021]
- Hayfa Tayeb [Inria, from Nov 2021]
- Arun Thangamani [Univ de Strasbourg, from Sep 2021]

Technical Staff

- Begatim Bytyqi [Inria, Engineer, from Feb 2021]
- Paul Cardosi [Inria, Engineer, until Oct 2021]
- Tiago Trevisan Jost [Univ de Strasbourg, Engineer, from Sep 2021]

Interns and Apprentices

- Emilien Bauer [Univ de Strasbourg, until Aug 2021]
- Nicolas Chappe [École Normale Supérieure de Lyon, until Feb 2021]
- Anton Danilkin [Inria, from May 2021 until Jul 2021]
- Tom Hammer [Univ de Strasbourg, from Sep 2021]
- Garip Kusoglu [Univ de Strasbourg, until Aug 2021]
- David Nicolazo [Univ de Strasbourg, from Jun 2021 until Aug 2021]
- Ludovic Paillat [Univ de Strasbourg, from Jun 2021 until Aug 2021]
- Clement Rossetti [Inria, from Jun 2021 until Aug 2021]
- Louis Savary [Inria, from May 2021 until Aug 2021]

- Hayfa Tayeb [Univ de Strasbourg, from Feb 2021 until Sep 2021]
- Michel Tching [Univ de Strasbourg, from Jun 2021 until Aug 2021]

Administrative Assistant

- Ouiza Herbi [Inria]

External Collaborator

- Stéphane Genaud [Univ de Provence, from Aug 2021, HDR]

2 Overall objectives

The CAMUS team is focusing on developing, adapting and extending automatic parallelization and optimization techniques, as well as proof and certification methods, for the efficient use of current and future multicore processors.

The team's research activities are organized into four main issues that are closely related to reach the following objectives: performance, correctness and productivity. These issues are: static parallelization and optimization of programs (where all statically detected parallelisms are expressed as well as all "hypothetical" parallelisms which would be eventually taken advantage of at runtime), profiling and execution behavior modeling (where expressive representation models of the program execution behavior will be used as engines for dynamic parallelizing processes), dynamic parallelization and optimization of programs (such transformation processes running inside a virtual machine), and finally program transformation proofs (where the correctness of many static and dynamic program transformations has to be ensured).

3 Research program

The various objectives we are expecting to reach are directly related to the search of adequacy between the software and the new multicore processors evolution. They also correspond to the main research directions suggested by Hall, Padua and Pingali in [57]. Performance, correctness and productivity must be the users' perceived effects. They will be the consequences of research works dealing with the following issues:

- Issue 1: Static Parallelization and Optimization
- Issue 2: Profiling and Execution Behavior Modeling
- Issue 3: Dynamic Program Parallelization and Optimization, Virtual Machine
- Issue 4: Proof of Program Transformations for Multicores

The development of efficient and correct applications for multicore processors requires stepping in every application development phase, from the initial conception to the final run.

Upstream, all potential parallelism of the application has to be exhibited. Here static analysis and transformation approaches (issue 1) must be performed, resulting in *multi-parallel* intermediate code advising the running virtual machine about all the parallelism that can be taken advantage of. However the compiler does not have much knowledge about the execution environment. It obviously knows the instruction set, it can be aware of the number of available cores, but it does not know the actual available resources at any time during the execution (memory, number of free cores, etc.).

That is the reason why a "virtual machine" mechanism will have to adapt the application to the resources (issue 3). Moreover the compiler will be able to take advantage only of a part of the parallelism induced by the application. Indeed some program information (values of the variables, accessed memory addresses, etc.) being available only at runtime, another part of the available parallelism will have to be generated on-the-fly during the execution, here also, thanks to a dynamic mechanism.

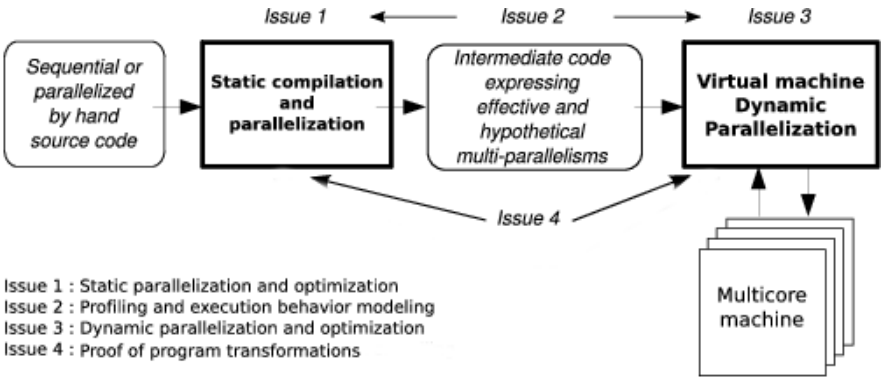

Figure 1: Steps for automatic parallelization on multicore architectures.

This on-the-fly parallelism extraction will be performed using speculative behavior models (issue 2), such models allowing to generate speculative parallel code (issue 3). Between our behavior modeling objectives, we can add the behavior monitoring, or profiling, of a program version. Indeed, the complexity of current and future architectures avoids assuming an optimal behavior regarding a given program version. A monitoring process will make it possible to select on-the-fly the best parallelization.

These different parallelization steps are schematized in figure 1.

Our project relies on the conception of a production chain for efficient execution of an application on a multicore architecture. Each link of this chain has to be formally verified in order to ensure correctness as well as efficiency. More precisely, it has to be ensured that the compiler produces a correct intermediate code, and that the virtual machine actually performs a parallel execution semantically equivalent to the source code: every transformation applied to the application, either statically by the compiler or dynamically by the virtual machine, must preserve the initial semantics. This must be proved formally (issue 4).

In the following, those different issues are detailed while forming our global, long term vision of what has to be done.

3.1 Static Parallelization and Optimization

Participants: Vincent Loechner, Philippe Clauss, Arthur Charguéraud, Bérenger Bramas

Static optimizations, from source code at compile time, benefit from two decades of research in automatic parallelization: many works address the parallelization of loop nests accessing multi-dimensional arrays, and these works are now mature enough to generate efficient parallel code [56]. Low-level optimizations, in the assembly code generated by the compiler, have also been extensively dealt with for single-core and require few adaptations to support multicore architectures. Concerning multicore specific parallelization, we propose to explore two research directions to take full advantage of these architectures: adapting parallelization to multicore architectures and expressing many potential parallelisms.

3.2 Profiling and Execution Behavior Modeling

Participants: Alain Ketterlin, Philippe Clauss

The increasing complexity of programs and hardware architectures makes it ever harder to characterize beforehand a given program's run time behavior. The sophistication of current compilers and the variety of transformations they are able to apply cannot hide their intrinsic limitations. As new abstractions like transactional memories appear, the dynamic behavior of a program strongly conditions its observed performance. All these reasons explain why empirical studies of sequential and parallel program executions have been considered increasingly relevant. Such studies aim at characterizing various facets of one or several program runs, *e.g.*, memory behavior, execution phases, etc. In some cases, such studies characterize more the compiler than the program itself. These works are of tremendous importance to

highlight all aspects that escape static analysis, even though their results may have a narrow scope, due to the possible incompleteness of their input data sets.

3.3 Dynamic Parallelization and Optimization, Virtual Machine

Participants: Philippe Clauss, Jens Gustedt, Alain Ketterlin, Bérenger Bramas

Dynamic parallelization and optimization has become essential with the advent of the new multicore architectures. When using a dynamic scheme, the performed instructions are not only dedicated to the application functionalities, but also to its control and its transformation, and so in its own interest. Behaving like a computer virus, such a scheme should rather be qualified as a "vitamin". It perfectly knows the current characteristics of the execution environment and owns some qualitative information thanks to a behavior modeling process (issue 2). It provides a significant optimization ability compared to a static compiler, while observing the evolution of the availability of live resources.

3.4 Proof of Program Transformations for Multicore Programs

Participants: Arthur Charguéraud, Alain Ketterlin

Our main objective consists in certifying the critical modules of our optimization tools (the compiler and the virtual machine). First we will prove the main loop transformation algorithms which constitute the core of our system.

The optimization process can be separated into two stages: the transformations consisting in optimizing the sequential code and in exhibiting parallelism, and those consisting in optimizing the parallel code itself. The first category of optimizations can be proved within a sequential semantics. For the other optimizations, we need to work within a concurrent semantics. We expect the first stage of optimization to produce data-race free code. For the second stage of optimization we will first assume that the input code is data-race free. We will prove those transformations using Appel's concurrent separation logic [58]. Proving transformations involving programs which are not data-race free will constitute a longer term research goal.

4 Application domains

Computational performance being our main objective, our target applications are characterized by intensive computation phases. Such applications are numerous in the domains of scientific computations, optimization, data mining and multimedia. In particular, several members of the team have contributed to high-performance code for numerical simulation of differential equations.

Applications involving intensive computations are necessarily high energy consumers. However this consumption can be significantly reduced thanks to optimization and parallelization. Although this issue is not our prior objective, we can expect some positive effects for the following reasons:

- Program parallelization tries to distribute the workload equally among the cores. Thus an equivalent performance, or even a better performance, to a sequential higher frequency execution on one single core, can be obtained.
- Memory and memory accesses are high energy consumers. Lowering the memory consumption, lowering the number of memory accesses and maximizing the number of accesses in the low levels of the memory hierarchy (registers, cache memories) have a positive consequence on execution speed, but also on energy consumption.

5 Highlights of the year

5.1 Awards

The paper *Specification and Verification of a Transient Stack*, by Alexandre Moine, Arthur Charguéraud, François Pottier received a **Distinguished Paper Award** for the conference *Certified Programs and Proofs* (CPP'22) [16]. Three papers out of 24 received this award.

6 New software and platforms

In 2021, we have made the following software contributions.

6.1 New software

6.1.1 SPETABARU

Name: SPEculative TAsk-BAsed RUntime system

Keywords: HPC, Parallel computing, Task-based algorithm

Functional Description: SPETABARU is a task-based runtime system for multi-core architectures that includes speculative execution models. It is a pure C++11 product without external dependency. It uses advanced meta-programming and allows for an easy customization of the scheduler. It is also capable to generate execution traces in SVG to better understand the behavior of the applications.

URL: https://gitlab.inria.fr/bramas/spetabaru

Contact: Bérenger Bramas

6.1.2 TRAHRHE

Name: Trahrhe expressions and applications in loop optimization

Keywords: Polyhedral compilation, Code optimisation, Source-to-source compiler

Functional Description: This software includes a mathematic kernel for computing Trahthe expressions related to iteration domains, as well as extensions implementing source-to-source transformations of loops for applying optimizations based on Trahrhe expressions.

URL: https://webpages.gitlabpages.inria.fr/trahrhe

Contact: Philippe Clauss

6.1.3 TLC

Name: TLC Coq library

Keywords: Coq, Library

Functional Description: TLC is a general purpose Coq library that provides an alternative to Coq's standard library. TLC takes as axiom extensionality, classical logic and indefinite description (Hilbert's epsilon). These axioms allow for significantly simpler formal definitions in many cases. TLC takes advantage of the type class mechanism. In particular, this allows for common operators and lemma names for all container data structures and all order relations. TLC includes the optimal fixed point combinator, which can be used for building arbitrarily-complex recursive and co-recursive definitions. Last, TLC provides a collection of tactics that enhance the default tactics provided by Coq. These tactics help constructing more concise and more robust proof scripts.

URL: http://www.chargueraud.org/softs/tlc/

Contact: Arthur Charguéraud

6.1.4 CFML

Name: Interactive program verification using characteristic formulae

Keywords: Coq, Software Verification, Deductive program verification, Separation Logic

Functional Description: The CFML tool supports the verification of OCaml programs through interactive Coq proofs. CFML proofs establish the full functional correctness of the code with respect to a specification. They may also be used to formally establish bounds on the asymptotic complexity of the code. The tool is made of two parts: on the one hand, a characteristic formula generator implemented as an OCaml program that parses OCaml code and produces Coq formulae, and, on the other hand, a Coq library that provides notations and tactics for manipulating characteristic formulae interactively in Coq.

URL: http://www.charguer aud.org/softs/cfml/

Contact: Arthur Charguéraud

Participants: Arthur Charguéraud, Armaël Guéneau, Francois Pottier

7 New results

7.1 Improvements of the C programming language

Participants: Jens Gustedt.

For the upcoming version of the C standard, C23, we already contributed with a large number of proposals that are already integrated in the working draft or that have been presented to ISO JTC1/SC22/WG14 [24] [25] [27] [29] [31] [41] [42] have been seen favorable and have good chances to be adopted [17] [34] [39] [26] [32] [33] [35] [49] [50] [51] [23] [28] [30] [32] [38] [40] [43] [44] [46] [47]. This concerns for example adjustment of the syntax and keywords [39, 34], an update of the admissible representations of integers [36, 44], clearer specification of synchronization between threads and other library features [37, 33], a concept of a `defer` keyword similar to the Go language [15, 45], and a multi-stage integration of lambdas and other features for type-generic programming [43]. Two other proposals are more long term and concern a possible adjustment between C and C++ [18] and the prospective new technical specification TS 6010 [19] for a sound and verifiable memory model.

7.2 Ordered Read-Write Locks for applications

Participants: Jens Gustedt.

In the context of a project with ICube's Mécaflu team we use our implementation of Ordered Read-Write Locks (ORWL) [4] to integrate parallelism into an already existing Fortran application that computes floods in the geographical region that is subject to the study. That parallelization has been achieved by using ORWL on a process level. It connects several executables that run the legacy code in separate processes and is able to launch such a stiched execution in shared memory and distributed contexts. As a first step to this goal it has been necessary to design a specific decomposition of geological data [48] and to implement a preconditioning [52].

7.3 Ionic Models Code Generation for Heterogeneous Architectures

Participants: Jens Gustedt, Arun Thangamani, Tiago Trevisan Jost, Vincent Loechner, Bérenger Bramas, Stéphane Genaud.

We participate to the research and development of a cardiac electrophysiology simulator in the context of the MICROCARD European project. We provide our optimizing compiler expertise to build a bridge from a high-level language convenient for ionic model experts (EasyML) to a code that will run on future exascale supercomputers. We aim to generate multiple versions of parallel parts to exploit the various parallel computing units that are available in the target architecture nodes (SIMD, multicore, GPUs, etc.).

The runtime infrastructure will rely on StarPU, in collaboration with the STORM team (Inria Bordeaux). The frontend is under development and it will be based on the MLIR extensible compiler. We already developed a Python script generating MLIR code from an EasyML ionic model description and integrated it in the openCARP project. We are currently working on the automatic vectorization of the ionic code. A publication on this topic is under preparation.

7.4 A Polyhedral Programmable Scheduler

Participants: Tom Hammer, Émilien Bauer, Vincent Loechner.

Scheduling is the central operation in the polyhedral compilation chain: finding the best execution order of loop iterations for optimizing the resulting code. Scheduling algorithms rely on the fundamental Farkas lemma to iteratively compute the schedule matrices associated to each statement of the input code.

We propose to develop a framework enabling the user to define a multi-objective optimization function, as a weighted composition of sub-objective functions. Those various sub-objectives include, among others:

- exposing outer parallelism for distribution and thread-level parallelization
- improving data locality
- exposing inner parallelism for SIMD and GPU parallelization
- etc.

The goal of this work is to allow polyhedral compilation experts to produce highly customizable scheduling algorithms in an efficient manner, without having to develop from scratch a new Farkas lemma based solver and the whole polyhedral compilation toolchain. It will be included in the PeriScop suite (OpenScop, CLooG, PIPLib, etc.). We will design a specific mini-language that allows the user to specify and prioritize its objectives.

This work has already started during Émilien Bauer's internship until June 2021. It will continue in Tom Hammer's internship, in connection with the MICROCARD project that will greatly benefit from the existence of such a scheduling algorithm generator.

7.5 Automatic Task-Based Parallelization using Source to Source Transformations

Participants: Bérenger Bramas, Stéphane Genaud, Garip Kusolgu.

We worked on a new approach to automatically parallelize any application written in an object-oriented language. The main idea is to parallelize a code as an HPC expert would do it using the task-based method. With this aim, we created a new source-to-source compiler on top of CLang-LLVM

called APAC. APAC is able to insert tasks in a source-code by evaluating data access and generating the correct dependencies. In 2021, we improved our compiler with a new method to automatically build performance models able to predict the execution time of the tasks. Using this method we are now able to activate only the tasks of a sufficient granularity.

7.6 A fast vectorized sorting implementation based on the ARM scalable vector extension (SVE)

Participants: Bérenger Bramas.

The release of the ARM scalable vector extension (SVE) changed radically the way we can vectorize. Indeed, SVE's interface is different in several aspects from the classical x86 extensions as it provides different instructions, uses a predicate to control most operations, and has a vector size that is only known at execution time. Therefore, using SVE opens new challenges on how to adapt algorithms, including the ones that are already well-optimized on x86. In this work, we ported a hybrid sort based on the well-known Quicksort and Bitonic-sort algorithms. We used a Bitonic sort to process small partitions/arrays and a vectorized partitioning implementation to divide the partitions. We relied on predicates and managed the non-static vector size. We tested the performance of our approach by sorting/partitioning integers, double floating-point numbers and key/value pairs of integers on a modern ARMv8.2 (A64FX) CPU, and we showed that our approach outperforms the GNU C++ sort by a speedup factor of 4 on average. This work has been published [11].

7.7 Shape- and scale-dependent coupling between spheroids and velocity gradients in turbulence

Participants: Bérenger Bramas.

The statistical studies of particles moving in a fluid is a widely used technic to study turbulences. We used this method to perform direct numerical simulations (DNS) of homogeneous isotropic turbulence and investigated the dynamics of different particle shapes at different scales in turbulence using a filtering approach. Bérenger Bramas collaborated on this project to design a particle interaction system able to work efficiently on large supercomputers [13].

7.8 Speculative Rewriting of Recursive Programs as Loop Candidates for Efficient Parallelization and Optimization Using an Inspector-Executor Mechanism

Participants: Salwa Kobeissi, Philippe Clauss.

Salwa Kobeissi defended her PhD thesis June the 24th, at the University of Strasbourg [20]. In her thesis, she introduces Rec2Poly, a framework for speculative rewriting of recursive programs as affine loops that are candidates for efficient optimization and parallelization. Rec2Poly seeks a polyhedral-compliant run-time control and memory behavior in recursions making use of an offline profiling technique. When it succeeds to model the behavior of a recursive program as affine loops, it can use the affine loop model to automatically generate an optimized and parallelized code based on the inspector-executor strategy for the next executions of the program. The inspector involves a light version of the original recursive program whose role is to collect, generate and verify run-time information that is crucial to ensure the correctness of the equivalent affine iterative code. The executor is composed of the affine loops that can be parallelized or even optimized using the polyhedral model.

7.9 Algebraic Loop Tiling

Participants: Clement Rossetti, Philippe Clauss.

We are currently developing a new approach for loop tiling, where tiles are no longer characterized by the sizes of their edges, but by their volumes, i.e., the number of embedded iterations. Algebraic tiling is particularly dedicated to parallel loops, where load balancing among the parallel threads is crucial for reaching high runtime performance.

Rectangular tiles of quasi-equal volumes, but of different shapes, are obtained thanks to mathematical computations based on the inversion of Ehrhart polynomials. The Trahrhe software (see Section 6.1.2) is dedicated to such computations, and to the automatic generation of header files in C, that can be used for the runtime computation of the algebraic tile bounds.

Clement Rossetti has developed a source-to-source wrapper of the Trahrhe software, that allows users to apply algebraic tiling to loop nests of their source codes, just by adding some dedicated pragmas. Although this software tool is robust, a number of extensions still need to be implemented, as the possibility of skewing the algebraic tile dimensions.

7.10 OptiTrust: Producing Trustworthy High-Performance Code via Source-to-Source Transformations

Participants: Arthur Charguéraud, Begatim Bytyqi.

Arthur Charguéraud obtained Inria funding for an "exploratory action" (Sep 2019 - Aug 2022). The aim of the project is to develop a framework for producing trustworthy high-performance code, starting from a high-level description of an algorithm that implements, *e.g.*, a numeric simulation. In 2021, we completed the set up of the framework, and completed the first major case study: a high-performance parallel implementation of a particle-in-cell algorithm used for plasma simulations. The naive implementation of the simulation consists of about 250 lines of C code. The optimization script consists of about 150 lines of OCaml code. The execution of that script produces a highly optimized code, essentially equivalent to the one that had been implemented by hand a few years ago [55]. A paper is under preparation.

7.11 A New Syle of Operational Semantics for Nondeterministic Languages

Participants: Arthur Charguéraud.

Arthur Charguéraud worked with collegues from MIT (Adam Chlipala and two of his students, Andres Erbsen and Samuel Gruetter) on a new style for describing operational semantics, particularly well-suited for nondeterministic languages. This technique introduces judgments that relate starting states to sets of outcomes, rather than to individual outcomes. Thus, a single derivation of these semantics for a particular starting state and program describes all possible nondeterministic executions, whereas in traditional small-step and big-step semantics, each derivation only talks about one single execution. We demonstrate how this restructuring allows for straightforward modeling of languages featuring both nondeterminism and undefined behavior. Specifically, our semantics inherently assert safety, *i.e.*, they guarantee that none of the execution branches gets stuck, while traditional semantics need either a separate judgment or additional error markers to specify safety in the presence of nondeterminism. Applications presented include proofs of type soundness for lambda calculi, mechanical derivation of reasoning rules for program verification, and a forward proof of compiler correctness for terminating but potentially nondeterministic programs. All results are formalized in Coq. Details are described in a draft paper [22].

7.12 Formal Verification of a Transient Data Structure

Participants: Arthur Charguéraud.

Between March and August 2021, Alexandre Moine's M2 internship was co-supervised by Arthur Charguéraud and François Pottier (Inria Paris, team Cambium). Alexandre used CFML2 (Charguéraud's implementation of Separation Logic inside the Coq proof assistant) to specify and verify the functional correctness and time complexity of a transient stack. A transient data structure is a package of an ephemeral data structure, a persistent data structure, and fast conversions between them. The transient stack studied by Alexandre is a scaled-down version of Sek, a general-purpose sequence data structure implemented in OCaml. Internally, it relies on fixed-capacity arrays, or chunks, which can be shared between several ephemeral and persistent stacks. Dynamic tests are used to determine whether a chunk can be updated in place or must be copied: a chunk can be updated if it is uniquely owned or if the update is monotonic. There are very few examples of verified transient data structures in the literature, let alone examples of transient data structures whose correctness and time complexity are verified, so we believe that this is an interesting contribution. This result has been accepted for presentation at the conference CPP 2022 [16] and has received one of the three Distinguished Paper Awards.

7.13 Raster Image Processing (RIP) Optimization

Participants: Vincent Loechner.

Our collaboration with the Caldera company on computer systems in charge of driving very wide printer farms and very fast digital printers has ended in 2020.

The final work of Paul Godard's PhD thesis on an out-of-core and out-of-place rectangular matrix transposition algorithm has been published in IEEE Transactions on Computers in November 2021 [12].

8 Partnerships and cooperations

8.1 European initiatives

8.1.1 MICROCARD

Participants: Vincent Loechner, Bérenger Bramas, Stéphane Genaud, Arun Thangamani, Tiago Trevisan Jost, Tom Hammer.

MICROCARD is a European research project to build software that can simulate cardiac electrophysiology using whole-heart models with sub-cellular resolution, on future exascale supercomputers. Vincent Loechner is the leader of *Work Package 6: Code generation for heterogeneous architectures* and co-leader of *Work Package 2: Task-based parallelization.*

We aim to build a bridge from a high-level model representation convenient for ionic model experts to an optimized implementation that exploits both target architecture resources and properties of the scientific problem (computation patterns, resilience to approximation). We will design a compiler infrastructure to translate an equational formulation extended with domain-specific information to a code that aims to be efficient in both execution time and energy dissipation.

- Funding: EuroHPC and ANR (France)

  This project is funded by the European High-Performance Computing Joint Undertaking EuroHPC (JU) under grant agreement No 955495. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Italy, Germany, Austria, Norway, Switzerland.

EuroHPC projects are for one half funded by the EuroHPC Joint Undertaking, and for the other half by the national funding agencies of the project partners.

- Start: April 2021
- End: September 2024
- Coordinator: Mark Potse (Univ. Bordeaux and CARMEN team)
- Partners: Univ. Bordeaux, Univ. Strasbourg, Inria, Karlsruhe Institute of Technology, Zuse Institute Berlin, Università della Svizzera italiana, University of Pavia, Simula (Univ. of Oslo), and the companies: OROBIX (Italia), MEGWARE (Germany), Numericor GmbH (Austria)
- Website

8.1.2 TEXTAROSSA

Participants: Bérenger Bramas, Hayfa Tayeb.

This European project aims at achieving a broad impact on the High Performance Computing (HPC) field both in pre-exascale and exascale scenarios [14]. The TEXTAROSSA consortium will develop new hardware accelerators, innovative two-phase cooling equipment, advanced algorithms, methods and software products for traditional HPC domains as well as for emerging domains in High Performance Artificial Intelligence (HPC-AI) and High Performance Data Analytics (HPDA). We will focus on the scheduling of task-graphs under energy constraints and on porting scientific codes on heterogeneous computing nodes with FPGAs.

- Funding: EuroHPC and ANR (France)

  This project is funded by the European High-Performance Computing Joint Undertaking EuroHPC (JU) under grant agreement No 956831. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Italy, Germany, Spain, and Poland. EuroHPC projects are for one half funded by the EuroHPC Joint Undertaking, and for the other half by the national funding agencies of the project partners.
- Start: April 2021
- End: April 2024
- Coordinator: Massimo Celino (ENEA)
- Partners: Agenzia Nazionale per l'Energia, le Nuove Tecnologie e lo Sviluppo Economico Sostenibile (ENEA), Fraunhofer Gesellschaft Zur Förderung der Angewandten Forschung E.V. (FGH), Consorzio Interuniversitario per l'Informatica (CINI), Institut National de Recherche en Informatique et en Automatique (Inria), Bull SAS, E4 Computer Engineering SpA, Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (BSC), Instytut Chemii Bioorganicznej Polskiej Akademii Nauk (PSNC), Istituto Nazionale di Fisica Nucleare (INFN), Consiglio Nazionale delle Ricerche (CNR), In Quattro Srl.
- Website

8.2 National initiatives

8.2.1 ANR Vocal

Participants: Arthur Charguéraud.

The goal of the ANR Vocal project is to develop the first formally verified library of efficient general-purpose data structures and algorithms. It targets the OCaml programming language, which allows for fairly efficient code and offers a simple programming model that eases reasoning about programs. The library is readily available to implementers of safety-critical OCaml programs, such as Coq, Astrée, or Frama-C. It provides a number of essential building blocks needed to significantly decrease the cost of developing safe software. The project combines the strengths of three verification tools, namely Coq, Why3, and CFML. It uses Coq to obtain a common mathematical foundation for program specifications, as well as to verify purely functional components. It uses Why3 to verify a broad range of imperative programs with a high degree of proof automation. Finally, it uses CFML for formal reasoning about effectful higher-order functions and data structures making use of pointers and sharing.

- Funding: ANR
- Start: October 2015
- End: March 2021
- Coordinator: Jean-Christophe Filliâtre (LRI)
- Partners: team VALS (Université Paris Sud), team Gallium (Inria Paris), team DCS (Verimag), team Camus, TrustInSoft, and OCamlPro.
- Website of Vocal

8.2.2 ANR AUTOSPEC

Participants: Bérenger Bramas, Philippe Clauss, Stéphane Genaud, Garip Kusoglu.

The AUTOSPEC project aims to create methods for automatic task-based parallelization and to improve this paradigm by increasing the degree of parallelism using speculative execution. The project will focus on source-to-source transformations for automatic parallelization, speculative execution models, DAG scheduling, and the activation mechanisms for speculative execution. With this aim, the project will rely on a source-to-source compiler that targets the C++ language, a runtime system with speculative execution capabilities, and an editor (IDE) to enable compiler-guided development. The outcomes from the project will be open-source with the objective of developing a user community. The benefits will be of great interest both for developers who want to use an automatic parallelization method, but also for high-performance programming experts who will benefit from improvements of the task-based programming. The results of this project will be validated in various applications such as a protein complexes simulation software, and widely used open-source software. The aim will be to cover a wide range of applications to demonstrate the potential of the methods derived from this project while trying to establish their limitations to open up new research perspectives.

- Funding: ANR (JCJC)
- Start: October 2021
- End: September 2025
- Coordinator: Bérenger Bramas

9 Dissemination

9.1 Promoting scientific activities

Member of the conference program committees

- Philippe Clauss has been part of the program committees of: the 11th and 12th editions of the International Workshop on Polyhedral Compilation Techniques (IMPACT 2021 and IMPACT 2022); the 18th Annual IFIP International Conference on Network and Parallel Computing (IFIP NPC 2021).

9.1.1 Journal

Member of editorial boards

- Jens Gustedt has been the Editor-in-Chief of the journal Discrete Mathematics and Theoretical Computer Science (DMTCS), since October 2001.

Reviewer - reviewing activities

- Bérenger Bramas was reviewer for the PeerJ Computer Science, Journal of Parallel and Distributed Computing (JPDC), Journal of Computer Science and Technology (JCST), The Journal of Open Source Software (JOSS), Software: Practice and Experience (SPE), and Concurrency and Computation: Practice and Experience (CCPE).
- Arthur Charguéraud has been a reviewer for the Journal of Functional Programming (JFP) and the journal Transactions on Programming Languages and Systems (TOPLAS).
- Philippe Clauss has been a reviewer for the Journal of Experimental and Theoretical Artificial Intelligence, IEEE Transactions on Computers, Mathematics, the Journal of King Saud University, PLOS ONE.

9.1.2 Scientific expertise

- Jens Gustedt has been a member of the ISO/IEC working groups ISO/IEC PL1/SC22/WG14 and WG21 for the standardization of the C and C++ programming languages, respectively.

9.1.3 Research administration

- Jens Gustedt is the head of the ICPS team for the ICube lab. He is a member of the executive board of directors of the lab, responsible for the IT and CS policy and for the coordination between the lab and the Inria center.
- Bérenger Bramas is in charge of the scientific computing group (axe transverse calcul scientifique) of the ICube laboratory and initiated the ICube software collection.

9.2 Teaching - Supervision - Juries

9.2.1 Teaching

- Licence: Vincent Loechner, Algorithmics and programmation, 82h, L1, Université de Strasbourg, France
- Licence: Vincent Loechner, System administration, 40h, Licence Pro, Université de Strasbourg, France
- Licence: Vincent Loechner, System programming, 20h, L2, Université de Strasbourg, France
- Licence: Vincent Loechner, Parallel programming, 32h, L3, Université de Strasbourg, France

- Licence: Vincent Loechner, System administration, 40h, Licence Pro, Université de Strasbourg, France
- Master: Vincent Loechner, Real-time systems, 12h, M1, Université de Strasbourg, France
- Eng. School: Vincent Loechner, Parallel programming, 20h, Telecom Physique Strasbourg - 3rd year, Université de Strasbourg, France
- Master: Bérenger Bramas, Compilation and Performance, 24h, M2, Université de Strasbourg, France
- Master: Bérenger Bramas, Compilation, 24h, M1, Université de Strasbourg, France
- Licence: Philippe Clauss, Computer architecture, 18h, L2, Université de Strasbourg, France
- Licence: Philippe Clauss, Bases of computer architecture, 22h, L1, Université de Strasbourg, France
- Master: Philippe Clauss, Compilation, 84h, M1, Université de Strasbourg, France
- Master: Philippe Clauss, Real-time programming and system, 37h, M1, Université de Strasbourg, France
- Master: Philippe Clauss, Code optimization and transformation, 31h, M1, Université de Strasbourg, France
- Licence: Alain Ketterlin, Architecture des systèmes d'exploitation, L3 Math-Info, 38h, Université de Strasbourg, France
- Licence: Alain Ketterlin, Programmation système, L2 Math-Info, 60h, Université de Strasbourg, France
- Master: Alain Ketterlin, Preuves assistées par ordinateur, 18h, Université de Strasbourg, France
- Master: Alain Ketterlin, Compilation, 84h, Université de Strasbourg, France

9.2.2 Supervision

- PhD: Salwa Kobeissi, *Speculative Rewriting of Recursive Programs as Loop Candidates for Efficient Parallelization and Optimization Using an Inspector-Executor Mechanism* [20], advised by Philippe Clauss, defended in June 2021.
- PhD in progress: Clément Flint, *Efficient data compression for high-performance PDE solvers.*, advised by Philippe Helluy (TONUS), Stéphane Genaud, and Bérenger Bramas, since Nov 2020.
- PhD in progress: Garip Kusoglu, *Automatic task-based parallelization by source-to-source transformations*, advised by Stéphane Genaud and Bérenger Bramas, since Oct 2021.
- PhD in progress: Hayfa Tayeb, *Efficient scheduling of task-based applications under energy constraints*, advised by Abdou Guermouche (HiePACS) and Bérenger Bramas, since Nov 2021.
- PhD in progress: Arun Thangamani, *Code generation for heterogeneous architectures*, advised by Stéphane Genaud and Vincent Loechner, since Sept 2021.
- PhD in progress: Alexandre Moine, *Formal Verification of Space Bounds*, is co-advised by Arthur Charguéraud and François Pottier, at Inria Paris, since Oct 2021.

9.2.3 Juries

- Philippe Clauss was a reviewer for:
  - the PhD thesis of Nuno Miguel Nobre, defended on May 27, 2021, at the University of Manchester, UK.
  - the PhD thesis of Tiago Trevisan Jost, defended on Jul. 2, 2021, at the University of Grenoble Alpes.

9.3 Popularization

9.3.1 Castor Informatique Contest

Participants: Arthur Charguéraud.

Arthur Charguéraud is a co-organizer of the Concours Castor informatique. The purpose of the Concours Castor in to introduce pupils, from CM1 to Terminale, to computer sciences. 671,000 teenagers played with the interactive exercises in November and December 2021 .

9.3.2 Modern C Book

Participants: Jens Gustedt.

Jens Gustedt authored the book Modern C [5], which since the first publication of an online draft in 2016 has become one of the major references for the C programming language. He also is blogging about efficient programming, in particular about the C programming language.

10 Scientific production

10.1 Major publications

[1] U. A. Acar, V. Aksenov, A. Charguéraud and M. Rainey. 'Provably and Practically Efficient Granularity Control'. In: *PPoPP 2019 - Principles and Practice of Parallel Programming*. Washington DC, United States, Feb. 2019. DOI: 10.1145/3293883.3295725. URL: https://hal.inria.fr/hal-01973285.

[2] P. Clauss. 'Counting Solutions to Linear and Nonlinear Constraints Through Ehrhart Polynomials: Applications to Analyze and Transform Scientific Programs'. In: *ICS, International Conference on Supercomputing*. ACM International Conference on Supercomputing 25th Anniversary Volume. Munich, Germany, 2014. DOI: 10.1145/2591635.2667172. URL: https://hal.inria.fr/hal-01100306.

[3] P. Clauss, F. J. Fernández, D. Garbervetsky and S. Verdoolaege. 'Symbolic polynomial maximization over convex sets and its application to memory requirement estimation'. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17.8 (Aug. 2009), pp. 983–996. DOI: 10.1109/TVLSI.2008.2002049. URL: https://hal.inria.fr/inria-00504617.

[4] P.-N. Clauss and J. Gustedt. 'Iterative Computations with Ordered Read-Write Locks'. In: *Journal of Parallel and Distributed Computing* 70.5 (2010), 496–504. DOI: 10.1016/j.jpdc.2009.09.002. URL: https://hal.inria.fr/inria-00330024.

[5] J. Gustedt. *Modern C*. Manning, 27th Nov. 2019. URL: https://hal.inria.fr/hal-02383654.

[6] A. Jimborean, P. Clauss, J.-F. Dollinger, V. Loechner and M. Juan Manuel. 'Dynamic and Speculative Polyhedral Parallelization Using Compiler-Generated Skeletons'. In: *International Journal of Parallel Programming* 42.4 (Aug. 2014), pp. 529–545. URL: https://hal.inria.fr/hal-01003744.

[7] A. Ketterlin and P. Clauss. 'Prediction and trace compression of data access addresses through nested loop recognition'. In: *6th annual IEEE/ACM international symposium on Code generation and optimization*. Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization. Boston, United States: ACM, Apr. 2008, pp. 94–103. DOI: 10.1145/1356058.1356071. URL: https://hal.inria.fr/inria-00504597.

[8] A. Ketterlin and P. Clauss. 'Profiling Data-Dependence to Assist Parallelization: Framework, Scope, and Optimization'. In: *MICRO-45, The 45th Annual IEEE/ACM International Symposium on Microarchitecture*. Vancouver, Canada, Dec. 2012. URL: https://hal.inria.fr/hal-00780782.

[9] B. Pradelle, A. Ketterlin and P. Clauss. 'Polyhedral parallelization of binary code'. In: *ACM Transactions on Architecture and Code Optimization*. Special issue on high-performance and embedded architectures and compilers 8.4 (Jan. 2012), 39:1–39:21. DOI: 10.1145/2086696.2086718. URL: https://hal.inria.fr/hal-00664370.

[10] A. Sukumaran-Rajam and P. Clauss. 'The Polyhedral Model of Nonlinear Loops'. In: *ACM Transactions on Architecture and Code Optimization* 12.4 (Jan. 2016). DOI: 10.1145/2838734. URL: https://hal.inria.fr/hal-01244464.

10.2 Publications of the year

International journals

[11] B. Bramas. 'A fast vectorized sorting implementation based on the ARM scalable vector extension (SVE)'. In: *PeerJ Computer Science* (19th Nov. 2021). URL: https://hal.inria.fr/hal-03227631.

[12] P. Godard, V. Loechner and C. Bastoul. 'Efficient Out-of-core and Out-of-place Rectangular Matrix Transposition and Rotation'. In: *IEEE Transactions on Computers* 70.11 (Nov. 2021), p. 7. DOI: 10.1109/TC.2020.3030592. URL: https://hal.inria.fr/hal-02960539.

[13] N. Pujara, J.-A. Arguedas-Leiva, C. C. Lalescu, B. Bramas and M. Wilczek. 'Shape- and scale-dependent coupling between spheroids and velocity gradients in turbulence'. In: *Journal of Fluid Mechanics* 922 (13th July 2021). DOI: 10.1017/jfm.2021.543. URL: https://hal.inria.fr/hal-03436562.

International peer-reviewed conferences

[14] G. Agosta, D. Cattaneo, W. Fornaciari, A. Galimberti, G. Massari, F. Reghenzani, F. Terraneo, D. Zoni, C. Brandolese, M. Celino et al. 'TEXTAROSSA: Towards EXtreme scale Technologies and Accelerators for euROhpc hw/Sw Supercomputing Applications for exascale'. In: DSD 2021 - 24th Euromicro Conference on Digital System Design. Palermo / Virtual, Italy, 1st Sept. 2021. URL: https://hal.inria.fr/hal-03329640.

[15] J. Gustedt and R. C. Seacord. 'C language mechanism for error handling and deferred cleanup'. In: SAC 2021 -The 36th ACM/SIGAPP Symposium on Applied Computing. Virtual, South Korea: ACM, 22nd Mar. 2021. DOI: 10.1145/3412841.3442116. URL: https://hal.inria.fr/hal-03059076.

[16] A. Moine, A. Charguéraud and F. Pottier. 'Specification and Verification of a Transient Stack'. In: CPP 2022 - 11th ACM SIGPLAN International Conference on Certified Programs and Proofs. Philadelphia, United States, 17th Jan. 2022. DOI: 10.1145/3497775.3503677. URL: https://hal.inria.fr/hal-03472028.

Scientific books

[17] É. Alepins and J. Gustedt. *Unsequenced functions*. ISO TC1/SC22/WG14, 31st Dec. 2021, p. 11. URL: https://hal.inria.fr/hal-02952723.

[18] J. Gustedt. *A Common C/C++ Core Specification*. ISO TC1/SC22/WG14, 20th Jan. 2021, p. 629. URL: https://hal.inria.fr/hal-02952700.

[19] J. Gustedt, P. Sewell, K. Memarian, V. B. F. Gomes and M. Uecker. *A Provenance-aware Memory Object Model for C*. ISO/IEC TC1/SC22/WG14, 28th Mar. 2021, p. 89. URL: https://hal.inria.fr/hal-02957464.

Doctoral dissertations and habilitation theses

[20] S. Kobeissi. 'Speculative Rewriting of Recursive Programs as Loop Candidates for Efficient Parallelization and Optimization Using an Inspector-Executor Mechanism'. Université de Strasbourg, 24th June 2021. URL: https://tel.archives-ouvertes.fr/tel-03495816.

Reports & preprints

[21] M. Boileau, B. Bramas, E. Franck, R. Hélie, P. Helluy and L. Navoret. *Parallel kinetic scheme for transport equations in complex toroidal geometry*. 8th Nov. 2021. URL: https://hal.archives-ouvertes.fr/hal-02404082.

[22] A. Charguéraud, A. Chlipala, A. Erbsen and S. Gruetter. *CPS Semantics: Smoother Nondeterminism in Operational Semantics*. 9th June 2021. URL: https://hal.inria.fr/hal-03255472.

[23] J. Gustedt. *Add annotations for unreachable control flow*. 2826. ISO JCT1/SC22/WG14, 3rd Oct. 2021, p. 10. URL: https://hal.inria.fr/hal-03265590.

[24] J. Gustedt. *Add annotations for unreachable control flow (slides)*. 30th Aug. 2021. URL: https://hal.inria.fr/hal-03328557.

[25] J. Gustedt. *Enforce storage stability (slides)*. 30th Aug. 2021. URL: https://hal.inria.fr/hal-03328555.

[26] J. Gustedt. *Enforce storage stability: proposal for addition to TS 6010*. 2756. ISO JCT1/SC22/WG14, 20th June 2021, p. 8. URL: https://hal.inria.fr/hal-03265588.

[27] J. Gustedt. *Function literals and value closures (slides)*. 10th Mar. 2021. URL: https://hal.inria.fr/hal-03165736.

[28] J. Gustedt. *Function literals and value closures: proposal for C23*. N2736. ISO JCT1/SC22/WG14, 15th May 2021, p. 55. URL: https://hal.inria.fr/hal-03106767.

[29] J. Gustedt. *Improve type generic programming (slides)*. 10th Mar. 2021. URL: https://hal.inria.fr/hal-03165732.

[30] J. Gustedt. *Improve type generic programming: proposal for C23*. N2638. ISO JCT1/SC22/WG14, 15th May 2021, p. 82. URL: https://hal.inria.fr/hal-03106758.

[31] J. Gustedt. *Lvalue closures: (slides)*. 14th June 2021. URL: https://hal.inria.fr/hal-03259361.

[32] J. Gustedt. *Lvalue closures: proposal for C23*. N2737. ISO JCT1/SC22/WG14, 15th May 2021, p. 12. URL: https://hal.inria.fr/hal-03106930.

[33] J. Gustedt. *Make call_once mandatory*. 2840. ISO JCT1/SC22/WG14, 12th Oct. 2021. URL: https://hal.inria.fr/hal-03390558.

[34] J. Gustedt. *Make false and true first-class language features: proposal for C2x*. N2885. ISO JTC1/SC22/WG14, 31st Dec. 2021. URL: https://hal.inria.fr/hal-02167916.

[35] J. Gustedt. *Only reserve names of optional functions if necessary*. 2839. ISO JCT1/SC22/WG14, 11th Oct. 2021. URL: https://hal.inria.fr/hal-03390554.

[36] J. Gustedt. *Pointers and integer types*. 2822. ISO JCT1/SC22/WG14, 3rd Oct. 2021. URL: https://hal.inria.fr/hal-03363711.

[37] J. Gustedt. *Remove ATOMIC VAR INIT*. N2886. ISO JTC1/SC22/WG14, 31st Dec. 2021. URL: https://hal.inria.fr/hal-02167838.

[38] J. Gustedt. *Require exact-width integer type interfaces*. 2821. ISO JTC1/SC22/WG14, 3rd Oct. 2021. URL: https://hal.inria.fr/hal-03363699.

[39] J. Gustedt. *Revise spelling of keywords: proposal for C2x*. N2884. ISO JTC1/SC22/WG14, 31st Dec. 2021. URL: https://hal.inria.fr/hal-02167870.

[40] J. Gustedt. *Type inference for variable definitions and function returns: proposal for C23*. N2735. ISO JCT1/SC22/WG14, 15th May 2021, p. 23. URL: https://hal.inria.fr/hal-03106763.

[41] J. Gustedt. *Type inference for variables and functions (slides)*. 10th Mar. 2021. URL: https://hal.inria.fr/hal-03165731.

[42] J. Gustedt. *Type-generic lambdas: (slides)*. 14th June 2021. URL: https://hal.inria.fr/hal-03259337.

[43] J. Gustedt. *Type-generic lambdas: proposal for C23*. N2634. ISO JCT1/SC22/WG14, 12th Jan. 2021, p. 12. URL: https://hal.inria.fr/hal-03106919.

[44] J. Gustedt. *Types and sizes*. 2838. ISO JTC1/SC22/WG14, 13th Oct. 2021. URL: https://hal.inria.fr/hal-03363692.

[45] J. Gustedt and R. C. Seacord. *A simple defer feature for C*. N2895. ISO JTC1/SC22/WG14, 31st Dec. 2021. URL: https://hal.inria.fr/hal-03524565.

[46] J. Gustedt and M. Uecker. *Disambiguate the storage class of some compound literals: change request for C23*. 2819. ISO JCT1/SC22/WG14, 3rd Oct. 2021, p. 2. URL: https://hal.inria.fr/hal-03363683.

[47] J. Gustedt and M. Uecker. *Properly define blocks as part of the grammar: proposal for C23*. 2818. ISO JCT1/SC22/WG14, 3rd Oct. 2021, p. 3. URL: https://hal.inria.fr/hal-03363674.

[48] S. Hariri, S. Weill, J. Gustedt and I. Charpentier. *A balanced watershed decomposition method Application to HEC-RAS*. 5th June 2021. URL: https://hal.archives-ouvertes.fr/hal-03250815.

[49] R. C. Seacord, S. Downey, J. Gustedt and P. Bindels. *Identifier Syntax using Unicode Standard Annex 31*. 2836. ISO JCT1/SC22/WG14, 13th Oct. 2021, p. 18. URL: https://hal.inria.fr/hal-03404598.

[50] M. Uecker and J. Gustedt. *Indeterminate Values and Trap Representations*. 2772. ISO TC1/SC22/WG14, 12th July 2021, p. 11. URL: https://hal.inria.fr/hal-03408023.

[51] M. Uecker and J. Gustedt. *Wide Function Pointer Types for Pairing Code and Data*. 2787. ISO JCT1/SC22/WG14, 15th Oct. 2021. URL: https://hal.inria.fr/hal-03404607.

Other scientific publications

[52] S. Hariri, J. Gustedt, S. Weill and I. Charpentier. 'A Hybrid Breaching-Filling method for sink removal adapted to parallel hydrological simulations'. In: EGU 2021. En ligne, France, 2021. DOI: 10.5194/egusphere-egu21-7849. URL: https://hal.archives-ouvertes.fr/hal-03238119.

10.3 Other

Softwares

[53] [SW] J. Gustedt, *Code examples for the book Modern C*, 16th Sept. 2021. LIC: MIT License. HAL: ⟨hal-03345464⟩, URL: https://hal.inria.fr/hal-03345464.

[54] [SW] A. Moine, A. Charguéraud and F. Pottier, *Specification and Verification of a Transient Stack (Artifact)*, 9th Dec. 2021. LIC: MIT License. HAL: ⟨hal-03473197⟩, URL: https://hal.inria.fr/hal-03473197, VCS: https://gitlab.inria.fr/amoine/cfml-sek, SWHID: ⟨swh:1:dir:80952d7602e8e9614979f1b1f50fdad8a4c369f1;origin=https://hal.archives-ouvertes.fr/hal-03473197;visit=swh:1:snp:781fc4f58efbdfa6ea16006272793bf7c610d760;anchor=swh:1:rel:3df60f40567bfdf2b365cac2fe61c6bb38d94503;path=/⟩.

10.4 Cited publications

[55] Y. A. Barsamian, A. Charguéraud, S. A. Hirstoaga and M. Mehrenberger. 'Efficient Strict-Binning Particle-in-Cell Algorithm for Multi-Core SIMD Processors'. In: *Euro-Par 2018 - 24th International European Conference on Parallel and Distributed Computing*. Turin, Italy, Aug. 2018. DOI: 10.1007/978-3-319-96983-1_53. URL: https://hal.archives-ouvertes.fr/hal-01890318.

[56] C. Bastoul. 'Code Generation in the Polyhedral Model Is Easier Than You Think'. In: *PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques*. Juan-les-Pins, France, 2004, pp. 7–16. URL: https://hal.archives-ouvertes.fr/ccsd-00017260.

[57] M. Hall, D. Padua and K. Pingali. 'Compiler research: the next 50 years'. In: *Commun. ACM* 52.2 (2009), pp. 60–67. URL: http://doi.acm.org/10.1145/1461928.1461946.

[58] A. Hobor, A. W. Appel and F. Z. Nardelli. 'Oracle Semantics for Concurrent Separation Logic'. In: *ESOP*. 2008, pp. 353–367.