Ph.D. in Information Technology:
Final Dissertations

DEIB- Seminar Room
January 7th, 2016
2.00 pm

First Ph.D. presentation and discussion:
Dr. Federica PANELLA – XXVII Cycle
Operator Precedence Languages: Theory and Applications
Supervisor: Prof. **Matteo Pradella**

Abstract: Operator Precedence Languages (OPLs) were introduced in the 1960s by Robert Floyd to support deterministic and efficient parsing of context-free languages. Recently, interest in this class of languages has been renewed thanks to a few distinguishing properties that make them attractive for exploiting various modern technologies in two main contexts: automatic software verification techniques, as model checking, and parallel and incremental parsing of programming and data-description languages.
This thesis provides a complete theory of OPLs and investigates the properties that allow for their application in these different fields.
Along a first line of research, we complement the results on this class of languages that have been proved in the last half a century, which characterized them in terms of equivalent classes of grammars, recognizing automata and a Monadic second-order logic; the study of their algebraic properties, furthermore, has qualified them as the largest class of deterministic context-free languages enjoying closure under all main language operations (Boolean ones, concatenation, Kleene * and others), strictly including renowned families of formalisms as parentheses languages and Visibly Pushdown Languages (VPLs). In this dissertation we extend research on OPLs to the field of omega-languages, i.e., languages consisting of strings of infinite length, which can model the behavior of systems with never-ending computations (such as operating systems, control systems, web services). We introduce an automata and Monadic second-order logic-based characterization for this class of languages and we prove their closure properties and the decidability of the emptiness problem, showing that they admit a decidable model checking problem. Furthermore, we study logic formalisms simpler than Monadic second-order logic to define suitable subclasses of OPLs.
On a second line of investigation, this dissertation deals with a further property enjoyed by OPLs that is not exhibited by other families of deterministic context-free languages such as LR and LL, namely their local parsability. Local parsability means that parsing of any substring of a string according to a grammar depends only on information that can be obtained from a local analysis of the portion of the substring under processing and hence is not influenced by parsing of other substrings. The lack of this property implies that parsing algorithms for, e.g., LR and LL languages are inherently sequential and cannot

exploit the speedup achievable by a parallel execution on modern multi-core computing platforms: in fact, if an input string is split into several parts, analyzed in parallel by different processing nodes, the parsing actions may require communication among the different processors, with considerable additional overhead. This thesis studies and exploits the local parsability property of OPLs to enable efficient parallel parsing of data description languages (as, e.g., the JSON standard data format) and programming languages (as, e.g., Lua and JavaScript) and presents a schema for parallelizing also the lexical analysis phase. The algorithms for parallel parsing and lexing have been implemented in a prototype tool (PAPAGENO), which we validated with an extensive experimental campaign, showing that they achieve significant, near-linear speedups on modern multicore architectures, overcoming state of the art sequential parsers and lexers generated by, e.g., Bison and Flex. We exploit the local parsability property enjoyed by OPLs also for efficient parallel querying of large structured and semi-structured documents.

Second Ph.D. presentation and discussion:
Dr. Michele SCANDALE - XXVIII Cycle
Towards Improving Programmability of Heterogeneous Parallel Architectures
Supervisor: Prof. **Giovanni Agosta**

Abstract: Parallel Computing has been considered an effective approach to combine performance and power efficiency for a long time.
Starting from High Performance Computing (HPC) to modern embedded systems, the employment of heterogeneous parallel architectures is becoming the common case, since they provide a good tradeoff in terms of power efficiency.
The exascale objective for the next generation of HPC systems is constrained to a target power envelope ranging from 20MW to 30MW. The existing ``Green'' HPC systems are not yet able to reach the such power efficiency although they already employ modern heterogeneous parallel architectures.
Ultra-low-power hardware platforms are gaining an increasing traction, as they may represent the key component to allow future HPC systems to match the required power efficiency.
The programmability of such systems is a critical aspect that has an huge impact on the reachable power efficiency and the effort required to reach such target. Programming parallel architectures is a complex task, since many hardware features are directly exposed to the programmers.
Programming frameworks that try to hide such complexity exist, however they either provide only sub-optimal performance with respect to hand tuned implementations, or they are limited to specific application domains.
This dissertation tackles challenges related to the programmability of heterogenous parallel architectures, acting on both existing and future programming models and hardware architectures.

In particular, we present OpenCRun, an OpenCL runtime implementation supporting a range of platforms with very different architectures characteristics, such as X86 multicores and embedded parallel accelerators.
In the context of ultra-low-power architectures we report the joint effort between hardware and software developers towards the PULP platform, showing the benefits of selected ISA extensions and their compiler support to maximize the power efficiency.
Moreover, to improve functional and performance portability of OpenCL code between GPGPUs and embedded many-core accelerators with explicitly managed memory such as PULP and STHorm, we have proposed a code transformation technique, work-item coalescing, that bypasses the limitations of the embedded platforms, allowing code developed for GPGPU to be ported seamlessly, as well as a memory transfer optimization technique to tune the resulting code to improve performance.
Finally, to increase the abstraction level in a more radical way, leveraging Shared Virtual Memory that is expected to be available in future architectures, we have presented a method to transparently implement shared function pointers in heterogeneous platforms with two or more ISAs, a building block for enabling full C++ support across heterogeneous ISAs.
Indeed we presented a fallback solution to implement function calls from device to functions not available on the device itself. This mechanism is needed to enable the transparent support of C++, and to provide more flexibility to the programmers dealing with large and complex applications to be ported towards heterogeneous parallel accelerators.