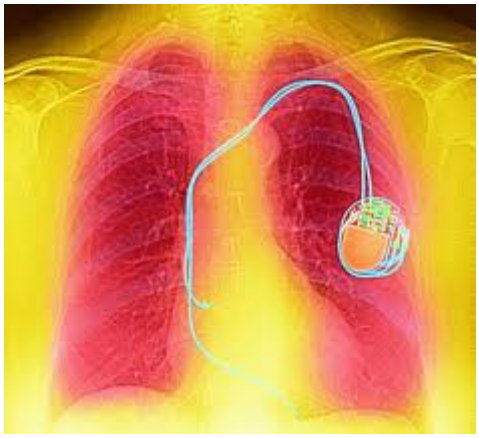
Energy Consumption Analysis and Verification using CiaoPP[1]

P. Lopez-Garcia[1,2], M.V. Hermenegildo[1,3], M. Klemen[1,3] and U. Liqat[1,3]

[1] IMDEA Software Institute
[2] Spanish Council for Scientific Research (CSIC)
[3] Universidad Politécnica de Madrid (UPM), Departamento de Inteligencia Artificial
{pedro.lopez,manuel.hermenegildo,maximiliano.klemen,umer.liqat}@imdea.org

Energy consumption and the environmental impact of computing technologies have become important concerns. There is increased demand for complex computing systems which have to operate on batteries or harvested energy, such as implantable/portable medical devices, space systems, mobile phones, and other *Internet of Things* devices. At the same time, energy consumption is also an important problem in high-performance computing and data centers.

Despite advances in power-efficient hardware, more energy savings can be achieved by improving the software. The estimation of the energy that program executions will consume is instrumental for program optimization and verification, and energy-aware software development in general. Performing such estimation statically (i.e., at compile-time, without running the program with concrete data) is much more useful than performing it dynamically, and is required in many interesting cases. However, this poses important challenges.

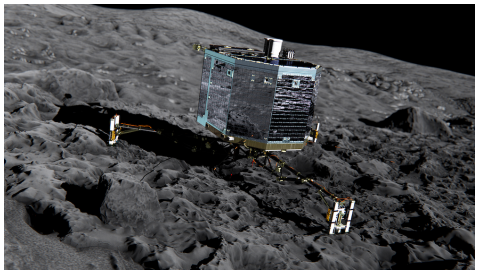


We have been developing an approach for: a) estimating such energy consumption statically in the form of *functions on the input data sizes of procedures* (and possibly other hardware-dependent parameters, such as clock frequency and voltage), and b) *using such functions for verifying and finding errors* with respect to a rich class of energy consumption specifications for programs. The approach has been implemented within the CiaoPP system. The purpose of this note is to provide a brief introduction to this work and the results obtained so far, as well as pointers to the corresponding papers, that provide the relevant details.

A fundamental observation is that determining program energy consumption requires the analysis of low-level program representations, i.e., at the level of Instruction Set Architecture (ISA), bytecodes, etc. This is because the necessary information regarding the energy-consuming operations really performed by the program is only present with sufficient accuracy at those levels. Our approach to the analysis and verification of energy consumption [22, 11, 16, 10, 8] is based

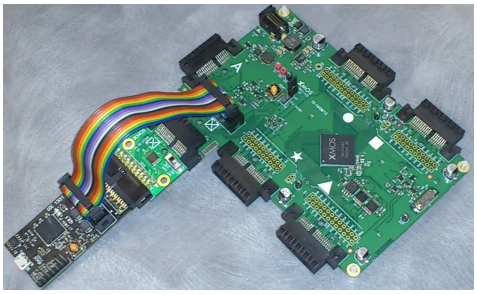
[1] The research leading to these results has received funding from the European Union 7th Framework Programme under grant agreement no 318337, ENTRA - Whole-Systems Energy Transparency, Spanish MINECO TIN2015-67522-C3-1-R *TRACES* project, and the Madrid M141047003 *N-GREENS* program.

on a transformation of these low-level programs into a set of Horn clauses [20, 6]. These Horn clauses encode the semantics of such low-level programs through different abstractions determined by the set of abstract domains used. Such abstract domains approximate properties that are instrumental for energy analysis, such as sized types [25], determinacy [12, 13], or non-failure [4, 1], as well as of course the cost / energy consumption itself. An interesting aspect of our approach to energy analysis (and to cost analysis in general), and which contrasts with previous approaches to cost analysis, is that it is based directly on abstract interpretation [26].



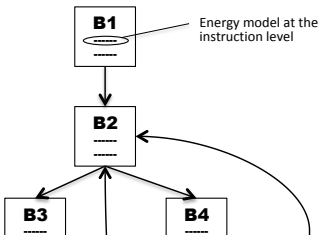
As is usual in abstract interpretation of logic programs, given such a set of Horn clauses, the objective of our analysis is to compute their abstract minimal model for each abstract domain or combination of domains. For this purpose we use the PLAI analyzer of the CiaoPP system [7]. PLAI computes this abstract minimal model using a top-down, memo table-based fixpoint algorithm, which can be seen as an extension of a highly optimized SLDT resolution engine with the abstract domains taking the traditional role of constraint domains in the logic. This extended abstract tabling algorithm includes optimizations for fixpoint acceleration such as dependency tracking or dynamic strongly-connected component detection. It is incremental and, in contrast to bottom-up algorithms (also available in CiaoPP), multi-variant (context sensitive). The result of the abstract model computation –the analysis– is a set of memo tables which store all the abstract call-success pairs that occur in the program, as well as dependencies between them. In the case of energy analysis these entries contain, for each procedure, and for each possible abstract call state and path, a function that returns the corresponding energy consumed by that procedure and class of calls, as a function of input data sizes.

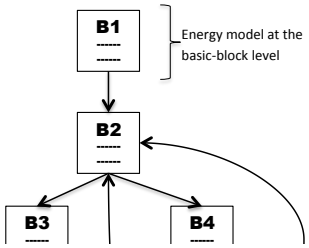
This work builds on our earlier work on cost analysis, initially developed for granularity control during automatic program parallelization [2, 18, 17], which was later extended for inferring both upper- and lower-bound functions [3, 5], and generalized for a wide class of *user-definable* resources [24, 23]. This configurability of the system allowed us to specialize it for execution time analysis [21] (using bytecode-level models obtained by regression), and is instrumental in the energy application for representing low-level energy models.



In particular, we have applied this approach and its implementation in CiaoPP to the energy analysis of (X)C programs running on the XMOS XS1-L architecture, both specialized for embedded programs. We have also performed a set of experiments comparing the energy analysis estimations for such programs to the actual energy consumption measured on the hardware, obtaining promising results [11, 10, 8, 9]. Embedded software developers can use this tool for determining values for program parameters that ensure meeting a given energy budget while minimizing losses in quality of service [16].

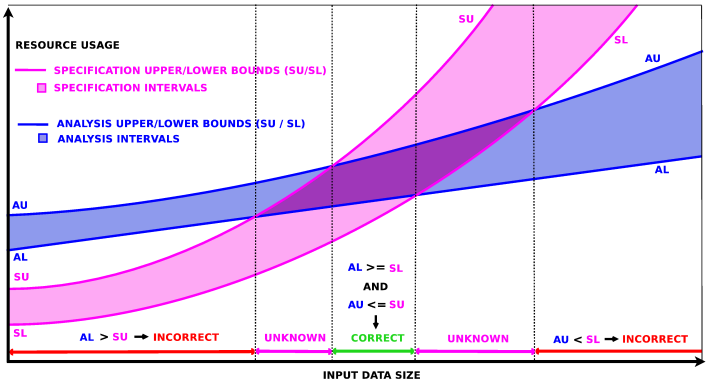
In these experiments several different energy models have been used. In [11, 10] the energy models used encoded energy consumption information for each ISA or LLVM instruction. Our results confirm that energy models at lower levels (e.g. ISA) are more precise than at higher levels (e.g., LLVM), but also that at lower levels more program structure and data structure information is lost, which can imply a loss of accuracy in the analysis. In our results the LLVM level emerges as a good compromise.





We have also explored using models of the energy consumption of whole blocks (i.e., block-level energy models) [8, 9]. Our approach combines static and dynamic (profiling-based) techniques for the inference of safe energy bounds of blocks. The dynamic technique uses an evolutionary algorithm to determine bounds on the energy consumption of each basic block. The CiaoPP static analyzer is then used to combine the energy values obtained for such blocks according to the program control flow, and produce energy consumption bounds of the whole program.

Our tools also perform verification and (performance) error detection following the CiaoPP assertion model [7, 15]. To this end, the inferred abstract models of energy consumption are compared to the energy specifications. This can optionally be done during the analysis or after it. In our approach specifications can include both lower and upper bounds on energy usage, and they can express intervals within which energy usage is to be certified to be within such bounds. The bounds of the intervals can be given in general as functions on input data sizes. Our verification system can prove whether such energy usage specifications are met or not. It can also infer the particular conditions under which the specifications hold. To this end, these conditions are also expressed as intervals of functions of input data sizes, such that a given specification can be proved for some intervals but disproved for others. The specifications themselves can also include preconditions expressing intervals for input data sizes.



Note that the type of information to be inferred by the analysis is motivated by its final use: program optimisation, verification, helping energy-aware software developers to make design decisions, etc. For example, for optimization the analysis can infer probabilistic information [27]. For verification, safe approximations (upper and lower bounds) need to be inferred [14, 15, 16], which poses some challenging problems. When the goal of the analysis is helping developers make resource-related design decisions, what is really needed is information that helps identify the parts (e.g., procedures) of a program responsible for highest fractions of the total resource usage of its execution, or how such total resource usage is *distributed* over those parts. For example, procedures which have lower costs but which are called more often may be responsible for a larger part of the overall resource usage, so that their optimization may be most profitable.

We have further extended and generalized the CiaoPP resource analysis framework so that it can now be specialized for inferring such kinds of information statically [19] (i.e., for static profiling). We show that the framework is general and flexible enough to support a wide range of static cost analyses, including both "accumulated" cost and the "standard notion" of cost.

References

[1] F. Bueno, P. López-García, and M. V. Hermenegildo. Multivariant Non-Failure Analysis via Standard Abstract Interpretation. In *7th International Symposium on Functional and Logic Programming (FLOPS 2004)*, number 2998 in LNCS, pages 100–116, Heidelberg, Germany, April 2004. Springer-Verlag.

[2] S. K. Debray, N.-W. Lin, and M. V. Hermenegildo. Task Granularity Analysis in Logic Programs. In *Proc. 1990 ACM Conf. on Programming Language Design and Implementation (PLDI)*, pages 174–188. ACM Press, June 1990.

[3] S. K. Debray, P. López-García, M. V. Hermenegildo, and N.-W. Lin. Lower Bound Cost Estimation for Logic Programs. In *1997 International Logic Programming Symposium*, pages 291–305. MIT Press, Cambridge, MA, October 1997.

[4] S.K. Debray, P. López-García, and M. V. Hermenegildo. Non-Failure Analysis for Logic Programs. In *1997 International Conference on Logic Programming*, pages 48–62, Cambridge, MA, June 1997. MIT Press, Cambridge, MA.

[5] S.K. Debray, P. López-García, M. V. Hermenegildo, and N.-W. Lin. Estimating the Computational Cost of Logic Programs. In *Static Analysis Symposium, SAS'94*, number 864 in LNCS, pages 255–265, Namur, Belgium, September 1994. Springer-Verlag.

[6] Kim S. Henriksen and John P. Gallagher. Abstract Interpretation of PIC Programs through Logic Programming. In *SCAM'06: Proceedings of the Sixth IEEE International Workshop on Source Code Analysis and Manipulation*, pages 184–196. IEEE Computer Society, 2006.

[7] M. V. Hermenegildo, G. Puebla, F. Bueno, and P. Lopez-Garcia. Integrated Program Debugging, Verification, and Optimization Using Abstract Interpretation (and The Ciao System Preprocessor). *Science of Computer Programming*, 58(1–2):115–140, October 2005.

[8] U. Liqat, Z. Banković, P. Lopez-Garcia, and M. V. Hermenegildo. Inferring Energy Bounds Statically by Evolutionary Analysis of Basic Blocks. In *Workshop on High Performance Energy Efficient Embedded Systems (HIP3ES 2016)*, 2016. arXiv:1601.02800.

[9] U. Liqat, Z. Banković, P. Lopez-Garcia, and M. V. Hermenegildo. Inferring Energy Bounds Statically by Evolutionary Analysis of Basic Blocks. In *Pre-proceedings of the 27th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR'17)*, October 2017.

[10] U. Liqat, K. Georgiou, S. Kerrison, P. Lopez-Garcia, M. V. Hermenegildo, J. P. Gallagher, and K. Eder. Inferring Parametric Energy Consumption Functions at Different Software Levels: ISA vs. LLVM IR. In M. Van Eekelen and U. Dal Lago, editors, *Foundational and Practical Aspects of Resource Analysis: 4th International Workshop, FOPARA 2015, London, UK, April 11, 2015. Revised Selected Papers*, volume 9964 of *Lecture Notes in Computer Science*, pages 81–100. Springer, 2016.

[11] U. Liqat, S. Kerrison, A. Serrano, K. Georgiou, P. Lopez-Garcia, N. Grech, M. V. Hermenegildo, and K. Eder. Energy Consumption Analysis of Programs based on XMOS ISA-level Models. In Gopal Gupta and Ricardo Peña, editors, *Logic-Based Program Synthesis and Transformation, 23rd International Symposium, LOPSTR 2013, Revised Selected Papers*, volume 8901 of *Lecture Notes in Computer Science*, pages 72–90. Springer, 2014.

[12] P. López-García, F. Bueno, and M. V. Hermenegildo. Determinacy Analysis for Logic Programs Using Mode and Type Information. In *Proceedings of the 14th International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR'04)*, number 3573 in LNCS, pages 19–35. Springer-Verlag, August 2005.

[13] P. López-García, F. Bueno, and M. V. Hermenegildo. Automatic Inference of Determinacy and Mutual Exclusion for Logic Programs Using Mode and Type Information. *New Generation Computing*, 28(2):117–206, 2010.

[14] P. López-García, L. Darmawan, and F. Bueno. A Framework for Verification and Debugging of Resource Usage Properties. In M. V. Hermenegildo and T. Schaub, editors, *Technical Communications of the 26th Int'l. Conference on Logic Programming (ICLP'10)*, volume 7 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 104–113, Dagstuhl, Germany, July 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[15] P. Lopez-Garcia, L. Darmawan, F. Bueno, and M. V. Hermenegildo. Interval-Based Resource Usage Verification: Formalization and Prototype. In R. Pe na, M.V. Eekelen, and O. Shkaravska, editors, *Foundational and Practical Aspects of Resource Analysis. Second International Workshop FOPARA 2011, Revised Selected Papers*, volume 7177 of *Lecture Notes in Computer Science*, pages 54–71. Springer-Verlag, 2012.

[16] P. Lopez-Garcia, R. Haemmerlé, M. Klemen, U. Liqat, and M. V. Hermenegildo. Towards Energy Consumption Verification via Static Analysis. In *Workshop on High Performance Energy Efficient Embedded Systems (HIP3ES), arXiv:1501.03064*, 2015. arXiv:1512.09369.

[17] P. López-García and M. V. Hermenegildo. Efficient Term Size Computation for Granularity Control. In *International Conference on Logic Programming*, pages 647–661, Cambridge, MA, June 1995. MIT Press, Cambridge, MA.

[18] P. López-García, M. V. Hermenegildo, and S. K. Debray. A Methodology for Granularity Based Control of Parallelism in Logic Programs. *Journal of Symbolic Computation, Special Issue on Parallel Symbolic Computation*, 21(4–6):715–734, 1996.

[19] P. Lopez-Garcia, M. Klemen, U. Liqat, and M. V. Hermenegildo. A General Framework for Static Profiling of Parametric Resource Usage. *Theory and Practice of Logic Programming, 32nd Int'l. Conference on Logic Programming (ICLP'16) Special Issue*, 16(5-6):849–865, October 2016.

[20] M. Méndez-Lojo, J. Navas, and M. Hermenegildo. A Flexible (C)LP-Based Approach to the Analysis of Object-Oriented Programs. In *17th International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR 2007)*, number 4915 in Lecture Notes in Computer Science, pages 154–168. Springer-Verlag, August 2007.

[21] E. Mera, P. López-García, M. Carro, and M. V. Hermenegildo. Towards Execution Time Estimation in Abstract Machine-Based Languages. In *10th Int'l. ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP'08)*, pages 174–184. ACM Press, July 2008.

[22] J. Navas, M. Méndez-Lojo, and M. Hermenegildo. Safe Upper-bounds Inference of Energy Consumption for Java Bytecode Applications. In *The Sixth NASA Langley Formal Methods Workshop (LFM 08)*, pages 29–32, April 2008. Extended Abstract.

[23] J. Navas, M. Méndez-Lojo, and M. V. Hermenegildo. User-Definable Resource Usage Bounds Analysis for Java Bytecode. In *Proceedings of the Workshop on Bytecode Semantics, Verification, Analysis and Transformation (BYTECODE'09)*, volume 253 of *Electronic Notes in Theoretical Computer Science*, pages 65–82. Elsevier - North Holland, March 2009.

[24] J. Navas, E. Mera, P. López-García, and M. Hermenegildo. User-Definable Resource Bounds Analysis for Logic Programs. In *23rd International Conference on Logic Programming (ICLP'07)*, volume 4670 of *Lecture Notes in Computer Science*. Springer, 2007.

[25] A. Serrano, P. Lopez-Garcia, F. Bueno, and M. V. Hermenegildo. Sized Type Analysis for Logic Programs (technical communication). In T. Swift and E. Lamma, editors, *Theory and Practice of Logic Programming, 29th Int'l. Conference on Logic Programming (ICLP'13) Special Issue, On-line Supplement*, volume 13, pages 1–14. Cambridge U. Press, August 2013.

[26] A. Serrano, P. Lopez-Garcia, and M. V. Hermenegildo. Resource Usage Analysis of Logic Programs via Abstract Interpretation Using Sized Types. *Theory and Practice of Logic Programming, 30th Int'l. Conference on Logic Programming (ICLP'14) Special Issue*, 14(4-5):739–754, 2014.

[27] H. Soza, M. Carro, and P. López-García. Probabilistic Cost Analysis of Logic Programs: A First Case Study. In *XXXII Latin-American Conference on Informatics*, August 2006.