Energy Consumption Analysis and Verification by Transformation into Horn Clauses and Abstract Interpretation*

M. V. Hermenegildo  M. Klemen  U. Liqat
IMDEA Software Institute
and
Universidad Politécnica de Madrid (UPM), Departamento de Inteligencia Artificial

P. Lopez Garcia
IMDEA Software Institute
and
Spanish Council for Scientific Research (CSIC)

{manuel.hermenegildo,maximiliano.klemen,umer.liqat,pedro.lopez}@imdea.org

Extended Abstract

The static estimation of the energy consumed by program executions has applications in program optimization and verification, and is instrumental in energy-aware software development. We describe our approach for estimating such energy consumption statically (i.e., at compile-time, without running the program) in the form of functions on the input data sizes of procedures (and possibly other hardware-dependent parameters), and for using such functions for verifying and finding errors with respect to a rich class of energy consumption specifications for programs. We also present the implementation of this approach within the CiaoPP system.

Determining program energy consumption requires the analysis of low-level program representations, since sufficiently accurate energy models are only available at the instruction set architecture level or, with some precision loss, at intermediate levels (such as, e.g., LLVM). We have developed an approach to the analysis and verification of energy consumption [14, 8, 10, 7, 6] that is based on a transformation of instruction set architecture- (or LLVM-)level programs into Horn clauses [13, 4]. The Horn clauses encode the semantics of these low-level programs, at different levels of abstraction, as determined by different abstract domains. Such abstract domains approximate properties that are instrumental for energy analysis, such as sized types, determinacy, or non-failure, as well as the energy consumption itself [17]. The latter contrasts with previous approaches to cost analysis that are not based directly on abstract interpretation.

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 [5]. PLAI computes the abstract minimal model of a set of Horn clauses 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 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.
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), multivariant (context sensitive). The result of the abstract model computation—the analysis—is the set of memo tables which store all the abstract call-success pairs that occur in the program (and their dependencies). In the case of energy analysis this includes, 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 [1][11], and which was capable of inferring both upper- and lower-bound functions [3][2], for a wide class of user-definable resources [16][15]. This configureability of the system is instrumental in the energy application for representing low-level energy models. As mentioned before, this work was recently extended by us to be fully based on multivariant abstract interpretation using sized shapes (sized types) [17], and to perform static profiling [12].

For verification and (performance) error detection the inferred abstract models of energy consumption are compared to the energy specifications [9][5]. 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.

We also report on a prototype implementation of our approach within the CiaoPP system for the XC language and XS1-L architecture, with an emphasis on embedded programs. We provide experimental results from energy analysis of such programs at the instruction set architecture level and compare them to the actual energy consumption measured on the hardware. We also illustrate with an example how 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.

References


Energy Consumption Analysis and Verification by Transformation


