Interpreting Energy Profiles with CEGAR

Steven te Brinke
brinkes@ewi.utwente.nl

Introduction

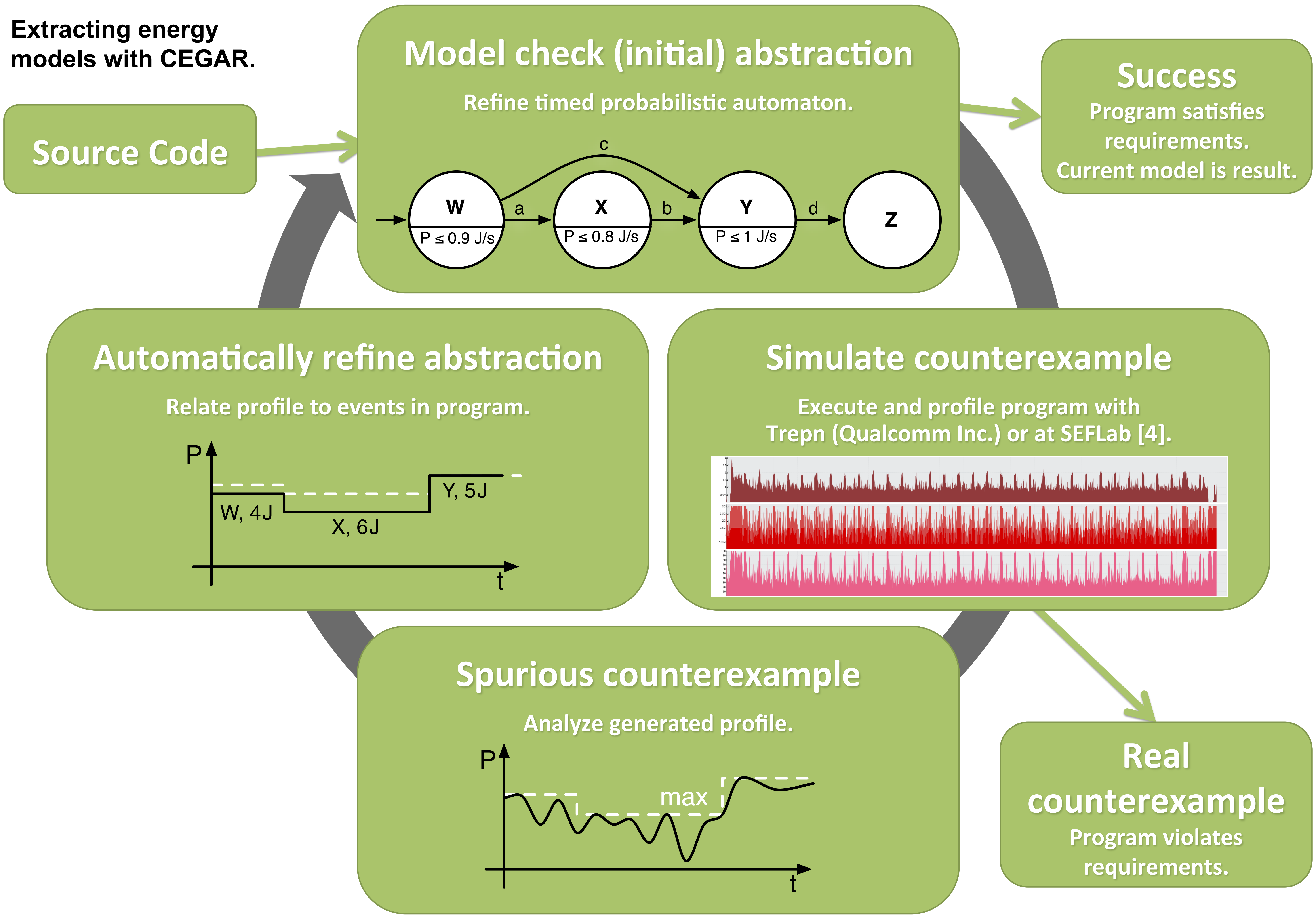
- Optimizing software energy consumption is important.
- Energy consumption of one component depends on the behavior of other components.
- Modeling resource consumption helps for analyzing dependencies and optimizing energy behavior.
- The Counterexample-Guided Abstraction Refinement (CEGAR) approach can automatically extract models from source code.

Problem Definition

- Existing CEGAR tools [1, 2, 3] do not consider energy consumption and timing.
- Existing CEGAR tools extract models from source code only, but energy information is usually not explicitly present in source code.
- Therefore, automatically extracted models lack energy information.

Work Summary

Extracting energy models with CEGAR.



Source Code

Model check (initial) abstraction

Refine timed probabilistic automaton.

Success

Program satisfies requirements. Current model is result.

Automatically refine abstraction

Relate profile to events in program.

Simulate counterexample

Execute and profile program with Trepn (Qualcomm Inc.) or at SEFLab [4].

Spurious counterexample

Analyze generated profile.

Real counterexample

Program violates requirements.

Expected Results

- Completely automated extraction of models from source code.
- These models contain sufficient energy information to optimize the energy consumption of the program.
- We optimize a media player based on these models to show that the models are useful in practice.

Conclusions

- Automatically extract models from source code.
- Extract energy profiles with Trepn from Android phones and with SEFLab equipment from desktop systems.
- Augment extracted models with energy information.
- Analyze the energy consumption of software components based on these models.
- Reduce the energy consumption of software implementations based on these models.

References

1. D. Beyer, T. A. Henzinger, R. Jhala, and R. Majumdar. The software model checker Blast: Applications to software engineering. *Int. J. Softw. Tools Technol. Transf.*, 9(5), 2007.
2. S. Chaki, E. Clarke, A. Groce, S. Jha, and H. Veith. Modular verification of software components in C. *Trans. Softw. Eng. (TSE)*, 30(6):388–402, June 2004.
3. E. Clarke, D. Kroening, N. Sharygina, and K. Yorav. SATABS: SAT-based predicate abstraction for ANSI-C. In *TACAS, volume 3440 of LNCS*. Springer, 2005.
4. M.A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, J. Visser. SEFLab: A lab for measuring software energy footprints. *Green and Sustainable Software (GREENS)*, pp. 30–37, 20 May 2013.

Acknowledgements

This work is carried out as part of the Energy Optimization Framework for Embedded Systems (ENOFES) project.

UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics and Computer Science
Department of Computer Science – Formal Methods and Tools Group
http://fmt.ewi.utwente.nl