CPALIEN: Shape Analyzer for CPAChecker*
(Competition Contribution)

Petr Muller and Tomáš Vojnar

FIT, Brno University of Technology, IT4Innovations Centre of Excellence, Czech Republic

Abstract. CPALIEN is a configurable program analysis framework instance. It uses an extension of the symbolic memory graphs (SMGs) abstract domain for shape analysis of programs manipulating the heap. In particular, CPALIEN extends SMGs with a simple integer value analysis in order to handle programs with both pointers and integer data. The current version of CPALIEN is an early prototype intended as a basis for a future research in the given area. The version submitted for SV-COMP'14 does not contain any shape abstraction, but it is still powerful enough to participate in several categories.

1 Verification Approach

CPALIEN is an analyzer of pointer manipulating programs written in the C language. It intends to handle industrial, often highly optimized code. CPALIEN is able to detect common memory manipulation errors like invalid dereferences, invalid deallocations, and memory leaks.

CPALIEN is an offspring of the successful Predator shape analyzer [1]. Predator implements a sound shape analysis of programs manipulating list-like data structures of various kinds. While Predator's ability to handle programs with complex lists is great (as witnessed by the tool winning gold medals in the appropriate categories in the first two SV-COMP competitions), we were unsuccessful with extending Predator to handle other data structures than lists and to also handle data other than pointers.

Therefore, we decided to redesign Predator's abstract domain of Symbolic Memory Graphs (SMGs) within the extensible framework of CPAChecker, another successful verification framework [4], which, however, so far lacked a support for shape analysis. Consequently, CPALIEN is implemented as an extension of CPAChecker and hence as an instance of the underlying Configurable Program Analysis (CPA) [3] framework.

Compared with the use of SMGs in Predator [2], the abstract domain of CPALIEN does not yet use any shape abstractions, which means the analysis will not terminate on programs building unbounded dynamic data structures (unless an error is found). On the other hand, CPALIEN combines usage of SMGs with a simple integer value analysis. Where possible, integer values are tracked explicitly for variables. When explicit values are not available, we infer information about value equality or nonequality from assumptions. The combination of this light-weight explicit integer value analysis and

* This work was supported by the Czech Science Foundation project 14-11384S and the EU/Czech IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070.

E. Ábrahám and K. Havelund (Eds.): TACAS 2014, LNCS 8413, pp. 395–397, 2014.

pointer analysis based on SMGs works well for enough test cases from the SV-COMP benchmark to get a positive score in categories where we participate.

The CPA framework allows one to merge the encountered states to reduce the generated state space. This feature is, however, not used in CPALIEN. To compute the covering relation, which is used by the high-level CPA reachability algorithm to determine the end of the state space search, CPALIEN uses the SMG join operation. CPALIEN also uses several specialized helper analyses provided by the CPAChecker framework to deal with certain specific tasks. These helper analyses are the *Location*, *CallStack*, and *FunctionPointer* CPAs.

2 Software Architecture

CPALIEN builds upon the CPAChecker framework for implementation, execution, and combination of instances of the CPA formalism. CPAChecker implements a reachability analysis algorithm over a generic CPA and also provides several other algorithms. CPALIEN is an implementation of a CPA instance, consisting of the abstract domain definition and the transfer relation between the states. Symbolic execution is driven by CPAChecker. CPAChecker also provides a C language parsing capability, wrapping a C parser present in the Eclipse CDT. Both CPAChecker and CPALIEN are written in Java.

3 Strengths and Weaknesses

A general strength of CPALIEN comes with implementation in the generic CPAChecker framework, offering a potential for the future in terms of combining the SMG-based shape analysis with other analyses.

Currently, CPALIEN is, however, mainly focused on heap manipulating programs as its integer value analysis plays just a supporting role without an ambition to handle harder problems. Moreover, CPALIEN is an early prototype, and it so far lacks any shape abstraction. Therefore, CPALIEN does not terminate on many of the benchmark test cases from the *Memory Safety* category for which CPALIEN is suited otherwise. For the *Heap Manipulation* category, the results are better: there are significantly more correct answers, with just a few timeouts and only a single false positive reported. Even the simple integer value analysis combined with the SMG domain managed to provide a correct answer for many test cases from the *Control Flow and Integer Value* category, especially those in the *Product Lines* sub-category.

Generally, the results correspond with the prototype status of the tool. Apart from the already mentioned missing abstraction, the tool still has many implementation issues. It also has deficiencies to handle some C language elements, like implicit type conversions. Another roadblock is CPALIEN's handling of external functions (functions with the body unavailable to the verifier). CPALIEN takes a stance that any unknown function can contain incorrect code, therefore the memory safety of programs calling unknown functions cannot be proved. An `UNKNOWN` answer is given for these cases. Therefore, CPALIEN's results could be improved by modeling the common C library functions, because many programs use them.

With these limitations being reflected by the results, we still argue that after their resolution, CPALIEN will form a promising base for further research on shape analysis

and its integration with other specialized analyses, providing heap analysis capabilities still missing in the CPAChecker ecosystem.

4 Tool Setup and Configuration

CPALIEN is available online at the project page:

`http://www.fit.vutbr.cz/research/groups/verifit/tools/cpalien/`

It is a modified version of the upstream CPAChecker, containing code not yet present in the upstream repository. For the participation in the competition, we have prepared a tarball. The only dependency needed to run CPALIEN is Java version 7.

For running the verifier, we have prepared a wrapper script to provide the output required by the competition rules. The script is run in the following way:

```
$ ./cpalien.sh target_program.c
```

Upon completion, a single line with the answer is provided. More information about the verification result, such as the error path, is provided in the `output` directory. The tool does not adhere to competition requirements with respect to property files: it does not allow a property file to be passed as a parameter. This was caused by our incorrect reading of the requirements. The property file is expected to be present in the same directory as the verification task.

CPALIEN participates in the *Heap Manipulation*, *Memory Safety* and *Control Flow and Integer Variable* categories. We opt out from the remaining ones.

5 Software Project and Contributors

CPALIEN is an extension of the CPAChecker project, building on the CPAChecker heavily. CPALIEN is developed by the VeriFIT [1] group at the Brno University of Technology. A significant part of the SMG code was contributed by Alexander Driemeyer from University of Passau, whom we would like to thank. CPAChecker is a project developed mainly by the Software Systems Lab[2] at the University of Passau. Both CPALIEN and CPAChecker are distributed under the Apache 2.0 license.

References

1. Dudka, K., Peringer, P., Vojnar, T.: Predator: A Practical Tool for Checking Manipulation of Dynamic Data Structures Using Separation Logic. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 372–378. Springer, Heidelberg (2011)
2. Dudka, K., Peringer, P., Vojnar, T.: Byte-Precise Verification of Low-Level List Manipulation. In: Logozzo, F., Fähndrich, M. (eds.) SAS 2013. LNCS, vol. 7935, pp. 215–237. Springer, Heidelberg (2013)
3. Beyer, D., Henzinger, T.A., Théoduloz, G.: Configurable Software Verification: Concretizing the Convergence of Model Checking and Program Analysis. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 504–518. Springer, Heidelberg (2007)
4. Beyer, D., Keremoglu, M.E.: CPACHECKER: A Tool for Configurable Software Verification. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 184–190. Springer, Heidelberg (2011)

[1] `http://www.fit.vutbr.cz/research/groups/verifit/`
[2] `http://www.sosy-lab.org/`