

# Free software tools for formal verification of computer programs

BY DAVID MENTRÉ

*Email:* dmentre@linux-france.org

*Version 1.5 - 2006-02-20*

We are now living in 2004<sup>1</sup>. We should no longer make software as in the sixties or seventies, with a few tests. We are now able to make software without **any** bugs. This is possible using specialised tools called *formal tools*. Such tools are able to match a computer program against a specification, i.e. a formal description of the expected behavior of the program. If the specification is correct and the formal verification can be done, then the program is guaranteed to be bug free. Of course, this is the ideal case and we are far from guaranteed bug free programs in the real world. But, as developer of free software, we should try to be as close as possible of this goal. As a first step, I list here free software tools that can help verification of computer programs.

## 1 Proof assistants

Proof assistants are computer programs that aids a human to prove things (so they are sometimes called *theorem provers*). Generally, they understand several formal logics with there rules and are able to apply those rules, automatically or guided by the human verifier. Such tools are at the core of the verification process.

### 1.1 Coq

Coq is a proof assistant environment.

- License: GNU LGPL 2.1
- Web site: <http://coq.inria.fr/>
- Debian packages: `coq coq-doc proofgeneral-coq`

### 1.2 ACL2

ACL2 is an environment where programs are described using an applicative subset of Common Lisp. Each function of the program entered in the environment is formally proven (termination, ...).

- License: GNU GPL
- Web site: <http://www.cs.utexas.edu/users/moore/acl2/>
- Debian packages: `acl2 acl2-books acl2-books-certs acl2-books-source acl2-doc acl2-emacs acl2-infix acl2-infix-source acl2-source`

### 1.3 Phox

PhoX is a proof assistant based on High Order logic and it is eXtensible. One of the principle of this proof assistant is to be as user friendly as possible and so to need a minimal learning time. The current version is still experimental but starts to be really usable. It is a good idea to try it and make comments to improve the final version.

- License: ??

---

1. Well, now we are in 2006. :-)

- Web site:  
[http://www.lama.univ-savoie.fr/sitelama/Membres/pages\\_web/RAFFALLI/phox.html](http://www.lama.univ-savoie.fr/sitelama/Membres/pages_web/RAFFALLI/phox.html)

## 1.4 HOL Light

HOL Light is a computer program to help users prove interesting mathematical theorems completely formally in higher order logic. It sets a very exacting standard of correctness, but provides a number of automated tools and pre-proved mathematical theorems (e.g. about arithmetic, basic set theory and real analysis) to save the user work. It is also fully programmable, so users can extend it with new theorems and inference rules without compromising its soundness.

- License: BSD like
- Web site: <http://www.cl.cam.ac.uk/users/jrh/hol-light/>

## 1.5 haRVey

Providing a high degree of automation to discharge proof obligations in (fragments of) first-order logic is a crucial activity in many verification efforts. Unfortunately, this is quite a difficult task.

On the one hand, reasoning modulo ubiquitous theories (such as lists, arrays, and Presburger arithmetic) is essential. On the other hand, to effectively incorporate this theory specific reasoning in boolean manipulations requires a substantial work.

The system haRVey implements a simple technique to cope with such difficulties whose aim is to check the validity of universally quantified formulae with arbitrary boolean structure modulo an equational theory.

The approach combines BDDs with refutation theorem proving. The former allows us to compactly represent the boolean structure of formulae, the latter to effectively mechanize the reasoning in equational theories.

- License: GNU LGPL 2.1
- Web site: <http://www.loria.fr/~ranise/haRVey/>

## 1.6 Brillant

Set of free software tools aiming at implementing the B method, for both software and hardware.

- License: GNU LGPL
- Web site: <https://gna.org/projects/brillant>

## 1.7 HOL

HOL 4 is the latest version of the HOL automated proof system for higher order logic: a programming environment in which theorems can be proved and proof tools implemented. Built-in decision procedures and theorem provers can automatically establish many simple theorems. An oracle mechanism gives access to external programs such as SAT and BDD engines. HOL 4 is particularly suitable as a platform for implementing combinations of deduction, execution and property checking.

- License: BSD like
- Web site: <http://hol.sourceforge.net/>

## 2 Model checkers

Model checkers are tool that verify all possible states of a formal model, i.e. a formal description of a system. Compared to proof assistant, they can be less powerful but easier to use.

## 2.1 NuSMV

NuSMV is a reimplement and extension of SMV, the first model checker based on BDDs. NuSMV has been designed to be an open architecture for model checking, which can be reliably used for the verification of industrial designs, as a core for custom verification tools, as a testbed for formal verification techniques, and applied to other research areas.

- License: GNU LGPL 2.1
- Web site: <http://nusmv.irst.itc.it/>

## 2.2 Murphi

Murphi also has a formal verifier based on explicit state enumeration. The verifier performs depth- or breadth-first search in the state graph defined by a Murphi description, storing all the states it encounters in a large hash table. When a state is generated that is already in the hash table, the search algorithm does not expand its successor states (they were expanded whenever the state was originally inserted in the table).

- License: BSD like
- Web site: <http://verify.stanford.edu/dill/murphi.html>

## 2.3 Mec 5

Mec 5 is a model-checker for finite AltaRica models, using a very expressive specification language (systems of fixpoint equations over finite relations with first-order quantifiers and equality testing).

- License: Public domain
- Web site: <http://altarica.labri.fr/Tools/Mec5/>

# 3 Tools to help verification of real programs

I have put under this category software that can be applied to real world programs (in language like C for example) to prove properties on them.

## 3.1 Why

Why is a verification conditions generator (VCG) back-end for other verification tools. It understands ML, C and Java languages (with the help of other programs).

- License: GNU GPL
- Web site: <http://why.lri.fr/>

## 3.2 CIL

CIL is a framework to analyse and manipulate C programs.

- License: BSD like
- Web site: <http://manju.cs.berkeley.edu/cil/>

## 3.3 Splint

Splint is a tool for statically checking C programs for security vulnerabilities and coding mistakes. With minimal effort, Splint can be used as a better lint. If additional effort is invested adding annotations to programs, Splint can perform stronger checking than can be done by any standard lint.

- License: GNU GPL

- Web site: <http://www.splint.org/>
- Debian packages: `splint splint-doc`

### 3.4 Cqual

Cqual is a type-based analysis tool that provides a lightweight, practical mechanism for specifying and checking properties of C programs. Cqual extends the type system of C with extra user-defined type qualifiers. The programmer adds type qualifier annotations to their program in a few key places, and Cqual performs qualifier inference to check whether the annotations are correct. The analysis results are presented with a user interface that lets the programmer browse the inferred qualifiers and their flow paths.

- License: GNU GPL
- Web site: <http://www.cs.umd.edu/~jfooster/cqual/>

### 3.5 CCured

CCured is a source-to-source translator for C. It analyzes the C program to determine the smallest number of run-time checks that must be inserted in the program to prevent all memory safety violations. The resulting program is memory safe, meaning that it will stop rather than overrun a buffer or scribble over memory that it shouldn't touch. Many programs can be made memory-safe this way while losing only 10–60% run-time performance (the performance cost is smaller for cleaner programs, and can be improved further by holding CCured's hand on the parts of the program that it does not understand by itself). Using CCured we have found bugs that Purify misses with an order of magnitude smaller run-time cost.

- License: BSD like
- Web site: <http://manju.cs.berkeley.edu/ccured/>

### 3.6 CHIC

CHIC is a modular verifier for behavioral compatibility checking of software and hardware components. The goal of CHIC is to be able to check that the interfaces for software or hardware components provide guarantees that satisfy the assumptions they make about each other. CHIC supports a variety of interface property specification formalisms.

- License: BSD like
- Web site: <http://www-cad.eecs.berkeley.edu/~arindam/Chic/>

### 3.7 Smatch!!!

Smatch is C source checker but mainly focused checking the Linux kernel code. It is based on the papers about the Stanford Checker.

Basically, Smatch uses a modified gcc to generate `.c.sm` files. The `.c.sm` files are piped through individual Smatch scripts that print out error messages.

For example, someone might want to write a Smatch script that looked for code that called `copy_to_user()` while the kernel was locked. If the script saw a place that called `lock_kernel()` then it would record the state as locked. If the script saw a place that called `unlock_kernel()` it would set the state to unlocked. If the state was locked and the script saw a place that called `copy_to_user()` the script would print out an error message.

- License: GNU GPL
- Web site: <http://smatch.sourceforge.net/>

### 3.8 Sparse

Sparse is a parsing and analysis library for the C language. One could put a number of different backends onto it; for example, a code-generation backend would turn it into a simple compiler. For the purposes of the kernel, however, the backend of interest is the analysis code which looks for various types of errors. The analyzer checks for quite a few different types of errors.

- License: OSL v1.1 ("Open Software License")
- Web site: none, however here is a [LWN Article describing sparse](#)
- Source: [bk://kernel.bkkbits.net/torvalds/sparse](http://kernel.bkkbits.net/torvalds/sparse)
- Mailing-list: <http://vger.kernel.org/vger-lists.html#linux-sparse>

### 3.9 Focal

Focal (formerly known as FoC) is a language for software-proof codesign. In Focal, code, specifications, and proofs are developed together in the same source files, using a novel object-oriented module system. The compiler analyses the dependencies in order to ensure the consistency of the source, then translates the code to Objective Caml, and the proofs to Coq.

- License: BSD like
- Web site: <http://modulogic.inria.fr/focal/download/>

## 4 Tools to make formal models

### 4.1 Alloy

The Alloy Analyzer is a tool developed by the Software Design Group for analyzing models written in Alloy, a simple structural modeling language based on first-order logic. The tool can generate instances of invariants, simulate the execution of operations (even those defined implicitly), and check user-specified properties of a model. Alloy and its analyzer have been used primarily to explore abstract software designs. Its use in analyzing code for conformance to a specification and as an automatic test case generator are being investigated in ongoing research projects.

- License: GNU GPL
- Web site: <http://alloy.mit.edu/>

## 5 Conclusion

It remains to test all those programs. A hard task that is not yet done.