Introducing the OKlibrary

Oliver Kullmann
Computer Science Department
Swansea University

Third Workshop on Formal and Automated Theorem Proving and Applications 2010
(Belgrade, 29.1.2010)

An "integrated research platform" (IRE) for (generalised) SAT

Introduction

Overview on the current state of the `OKlibrary`:

`http://www.ok-sat-library.org`

- the pre-alpha version will be released rather soon
- an attempt to build a research environment for all aspects related to **theoretical** and practical research on (generalised) SAT
- currently focusing on a convenient

  interactive environment.

The Maxima/Lisp level

Overview

External Sources

The build system

Planning and documentation

Conclusions and Outlook

Overview

1. The Maxima/Lisp level
2. Overview
3. External Sources
4. The build system
5. Planning and documentation
6. Conclusions and Outlook

Basic ideas of the Maxima/Lisp level

- “pseudo-code which runs”
- “procedural specification”
- exploring the whole world of generalised SAT, NP and beyond
- also for teaching
- it’s not fast but — that is a feature, not a bug.

The design goal is to ignore (as completely as possible)

efficiency concerns **as well as abstraction**
(not in the mathematical sense, but in the sense of abstract data types — no ADTs),

and to write small functions which directly communicate the underlying mathematical ideas.

The Maxima/Lisp level

Overview

External Sources

The build system

Planning and documentation

Conclusions and Outlook

Avoiding premature abstraction

- We provide “pseudo-code which runs”, where readability and precision is of utmost importance.
- So the code is treated as some sort of a “procedural specification”, to be used as the basis for all further developments.
- No abstraction in the form of “abstract data types” is provided at this level, but “the objects speak for themselves”.
- So this level is concrete and precise, and we try to take care explicitly of **all details**.

The whole world

- In the long run, we want to explore the whole world of generalised SAT, NP and beyond at this level.
- Hopefully the functionality provided at this level (together with the code(!)) is also useful for teaching.
- Finally, execution of the functions provided at this level is rather slow, emphasising the conceptual and specificational nature of this level of the `OKlibrary`.

The Maxima/Lisp level

Overview

External Sources

The build system

Planning and documentation

Conclusions and Outlook

Why Maxima?

1. open-source
2. general purpose (from discrete mathematics to mathematical physics)
3. corresponding to the fundamental mathematical level (set theory), replacing sets by lists.

Only a semi-strong community, some historical ballast, and weaknesses regarding the process of software development, but one can (and has to) live with that.

The Maxima/Lisp level

Overview

External Sources

The build system

Planning and documentation

Conclusions and Outlook

3 levels (a cumulative hierarchy)

Three programming language layers (adding more and more aspects):

1. the Maxima/Lisp level — the “set-theoretic level”
2. (the Axiom/Aldor level: adding
   1. polymorphism, generic programming
   2. abstract data types
   3. also fundamental data structures)
3. the C++ level: the full level, adding
   1. full control over algorithmic details (including construction and destruction of objects)
   2. distinction between compile-time and run-time.

Development in stages

- So we consider it as an advantage that certain programming languages cannot express (appropriately) certain aspects of programming.
- However we do not make a fetish out of it, as if “thou shall not consider those details”.
- Efficiency considerations are not “small details left to some employees” but are deeper properties which demand their own sphere.
- But this best at later stages of the development, when a clearer picture emerged.

The currently usable parts

- the Maxima/Lisp level
- various components at C++ level
- `ExternalSources`
- the documentation system
- the test system
- the build system
- source control.

Mission statement

An **integrated research environment**
for generalised SAT,
built together from powerful components
in the Unix/Linux tradition,
combined through the `OKplatform`,
with the `OKlibrary` as core.

On the C++ level

The `OKlibrary` started as a

generative library for generalised SAT solving.

Meanwhile the scope has broadened, but the C++ level still is of great importance (for the future):

1. many plans
2. lots of code
3. but yet postponed/delayed due to **C++ 09**.

Kind of a Linux distribution for SAT

There is a lot of interesting software out there, but

- it's hard to find
- hard to install
- often undocumented.

Now we have the means (our package is 700 MB anyway), we need to do it for us, and so we want to offer (all) external sources with automated (complete!) installation and additional documentation.

What is available

- C/C++ libraries
- various compiler
- programming support
- buildsystem components
- (databases)
- statistics and computer algebra
- (proof assistants and automated theorem proving)
- last, but not least, SAT.

Higher-order unit testing

Starting at the Maxima/Lisp level we have a unit test system

which is not, as usual, left to the back office, but is integral part of the library.

Generic test functionality is offered, which can be reused.

A similar (stronger) system is supplied at the C++ level.

Integration and application testing

An application test system is under construction.

Specialised to SAT solving, this will be also applicable for “external SAT solvers”:

1. specify a test level: “basic”, “full” or “extensive”
2. watch whether your solver survives.

The build system in general

Not just testing, but as many build-aspects as possible should be automated. So a build system is part of the platform.

- We use `make`.
- There are other shinier systems out there, but none seems to offer the generality we need.

A “holistic” library

I do not mean here the quasi-religious implications of some interpretations of “holism” (“The whole is more than the sum of its parts.”), but that

the library is “complete in itself”,

or, in other words, it is **completely clonable**. (ahem, in principle ...)

- We have already seen one example with the higher-order unit test system.
- Another aspect is that nearly all planning is “internalised” (available via the html documentation).

A strong source control system is needed. We use the most powerful open-source system, `Git`.

Summary

I. I hope you got an impression of the parts of the library which are currently usable by the uninitiated user.

II. For the years to come, I hope a strong system can be created, spanning the whole of the “generalised SAT universe”.

End