Advanced Documentation and Testing Tool (ADATT)

real-world formal documentation

Formal methods are great

- complexity of hard- and software is ever increasing
- formal methods are vital to deal with this
- I'm especially interested in interactive theorem proving (ITP)
- other formal methods important as well
- amazing feats have been achieved in recent years
  - very trustworthy formal models of hardware, protocols and programming languages
  - verified and verifying compilers of real-world languages
  - verified operating systems
  - ...
- formal methods tools are powerful and even have good GUIs

however, they are hardly used

Formal methods are hardly used

- even critical, very well funded projects hardly use interactive theorem proving
- distinction between automatic and interactive tools is blurred
- fully automated tools usually
  - are better accepted and more frequently used
  - but lack in expressive power and scalability

reasons for this lack of usage are some issues, but also prejudice against formal methods

Prejudice

Even highly skilled, clever computer scientists believe:

"Interactive theorem proving and formal methods are a form black magic: they are way to complicated for mere mortals and while huge gains are luring, you need to sell your soul for these gains."

Issues with formal methods

- math, logic, proofs, etc. are complicated
- formal methods experts are needed
- it takes a lot of time
- progress and benefits are hard to measure

hide formal methods

good tools can mitigate issues

Advanced Documentation and Testing Tool (ADATT)

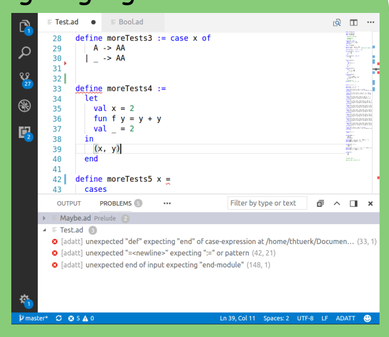
- I'm developing a tool to mitigate these issues
- programmers and test engineers write formal specification disguised as documentation and tests
- formal method experts extend it and deal with proofs
- time saved by synergy effects
- inspired by Lem, but ADATT focuses of partial specifications

ADATT ideas

- most bugs are found by
  - creating a formal specification
  - testing
  - simple, local formal-sanity checks
- deep, complicated proofs reveal comparably few bugs
- formal method experts only needed for such deep proofs
- programmers are able and willing to write formal specifications, if
  - they are disguised as documentation, pseudo-code, tests, ...
  - it is natural and comfortable
  - there is an immediate benefit for the programmer
- time is saved by
  - synergies between creating formal specifications, writing documentation and testing
  - improved communication between formal-method- and domain-experts
- progress-measures allow integration in development processes
  - coverage
  - simple statistics
  - keeping track of run tests, previous bugs, ...

ADATT look and feel

- ADATT looks like a compiler for a simple functional programming language with support for
  - writing documentation
  - writing tests, code contracts, properties ...
  - partial implementations
- ADATT can compile to
  - high-quality human readable documentation
  - executable specifications in common programming languages
  - specifications for common interactive theorem provers
  - code snippets in various programming languages
    - for conformance testing
    - import / export of data
    - ...
- good user experience is vital
  - good integration in common IDEs
  - good error and warning messages
  - simple semantics
  - language is easily readable



ADATT workflow

- start with writing natural language, informal documentation
- step by step add more formal content
  - function and type declaration, type signatures
  - test cases, invariants, code contracts
  - executable specifications
- adding formal content added must have immediate benefit
- progress is easily measurable
- good support for partial specifications is vital
- resulting executable specifications that can be used as
  - test oracle / reference implementation
  - formal model whose properties can be reasoned about
- distinction between documentation, tests and specifications blurred
- designers, programmers, test engineers and formal method experts can all work with different parts of an ADATT project

Thomas Türk
email: thomas@tuerk-brechen.de
http://www.thomas-tuerk.de/en