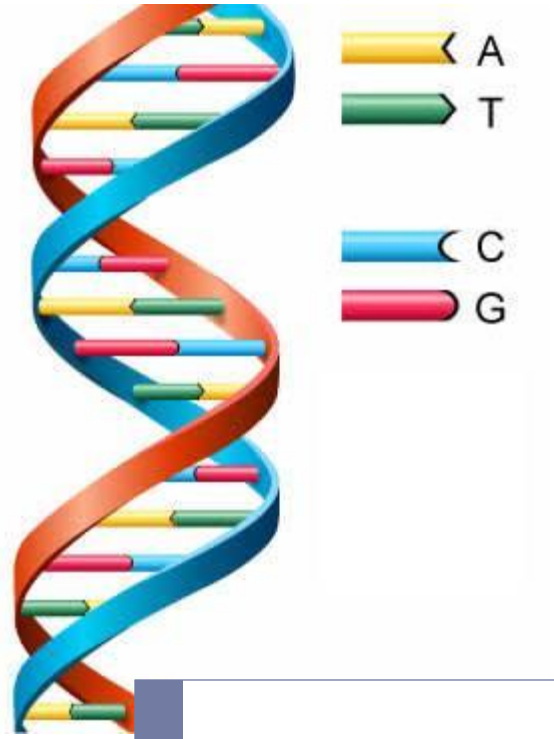


Mutation Testing for Erlang – what are the interesting invariants?

Hans Svensson – FP Workshop,
Chalmers 2010-06-02

The ideas in the presentation is joint "work" with John Hughes and Michał Pałka

Software Testing

- When have we tested enough?
- Are my properties covering all aspects of the program?
- Is this a god test-suite?
- …

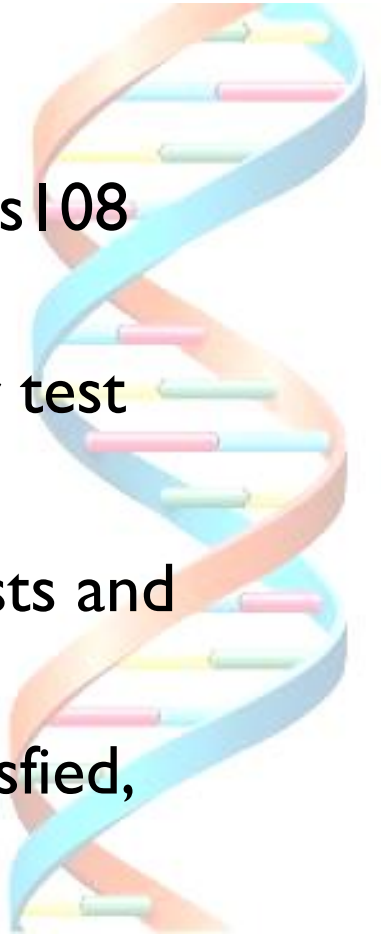
- Test-adequacy criteria is needed, on popular approach:

Mutation testing

Mutation Testing

- Simple idea: (Automatically) change the program into a similar program and see if the error is found by the tests
  - What changes?
  - How many programs to generate?
- Very active research topic for imperative languages (Java, C/C++, etc.)
- Not widely used for functional languages
  - Maybe for a good reason!?
  - We want to try it for Erlang

Mutation Testing in Practice



- 1. Select mutation operators (one paper repors 108 standard operators)
- 2. Generate all mutants for the program under test
- 3. Compile and run tests for all mutants
- 4. Filter out mutants that are not failing any tests and classify them into **equivalent** or **killable**
- If the all killable mutants are killed you are satisfied, otherwise you should improve your tests

- Very expensive!! Even for small programs the number of mutants quickly get out of hand…

Optimizing Mutation Testing

- Use *symbolic* mutations
  - Less compilation, but still computation heavy
- Use only a subset of the mutation operators
  - Which to choose is a widely studied topic
  - Random choice does not seem to be much worse
- Neither option is hard to implement/use in a mutation testing tool for Erlang, but the question remains:

  What are the interesting mutations?

Select interesting mutation operators

- Use the operators presented in literature
  - Arithmetic operator replacement
  - Statement deletion
  - Relational operator replacement
  - …
- Think hard
  - This presentation is part of the thinking hard strategy…
- Do some sort of experiment/analysis
  - Mining software repositories!?

Your input/ideas!?

- Is it a bad idea altogether?
- What is the main differences between mutation testing for say C/C++ and Erlang/Haskell?
- What interesting mutations are there?
- Other ways to find interesting mutations?

Thanks!