Seminar on Program Analysis and Software Testing – Introduction

Alexander Nutz Christian Schilling

Chair of Software Engineering, Department of Computer Science, University of Freiburg

October 31, 2016

Overview

Part 1: Basic terminology for many seminar topics

Part 2: Hints on reading a paper, giving a talk

Part 1, Terminology – Disclaimer

- Glossary
- "General ideas"
- Informal
- *One* terminology out of many

Questions about Programs

- Program analysis

  program behaviour?

- Program verification

  program behaviour $\overset{?}{\subseteq}$ specified behaviour

- Testing

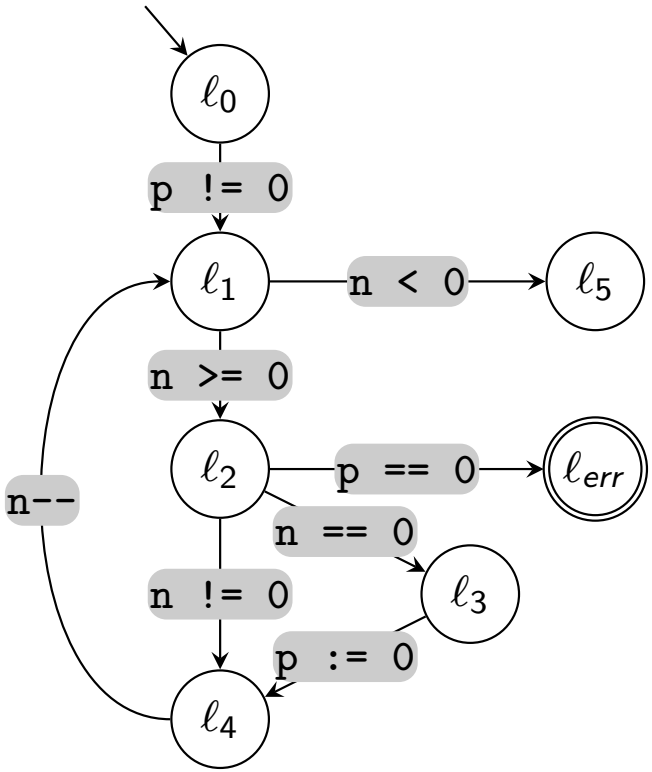
  Is there a failing test case?
  How good are my non-failing test cases?

Control Flow Graph

- Nodes: represent program locations, roughly: line numbers
- Edges: represent execution of a statement

Example program:

```
ℓ0:    assume p != 0;
ℓ1:    while(n >= 0)
       {
ℓ2:        assert p != 0;
           if(n == 0)
           {
ℓ3:            p := 0;
           }
ℓ4:        n--;
       }
```

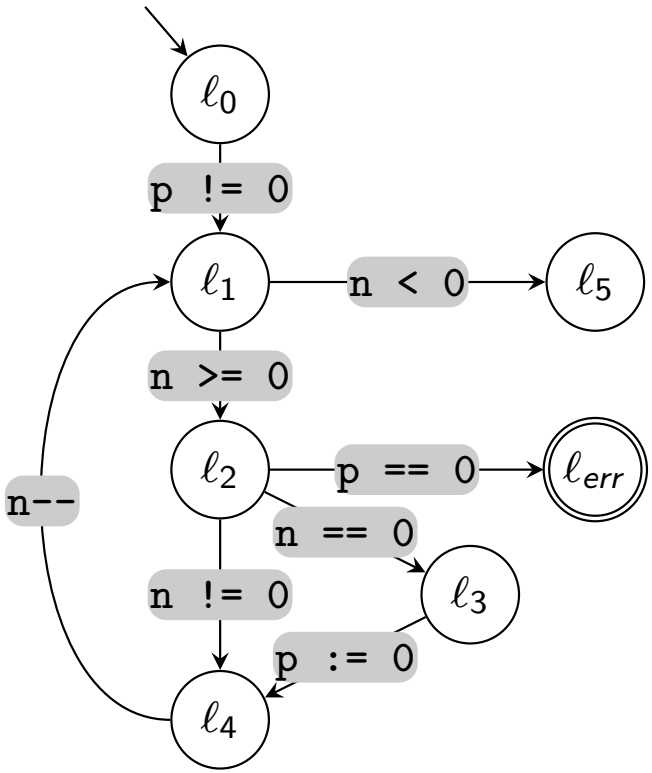


Example taken from poster by Matthias Heizmann on Ultimate Automizer for SV-COMP 2014

The assume statement

- Syntax: `assume` φ for some logical formula φ
- Semantics: If φ holds, continue execution; otherwise block.

```
ℓ0:   assume p != 0;
ℓ1:   while(n >= 0)
      {
ℓ2:       assert p != 0;
          if(n == 0)
          {
ℓ3:           p := 0;
          }
ℓ4:       n--;
      }
```



State

- A program's *state* (at some point of execution): valuation of variables. For example:

$$x \mapsto 4, y \mapsto 3.8, pc \mapsto \ell_{13}$$

  Other names: *concrete* or *explicit* state

- Transition system/Kripke structure is a graph with

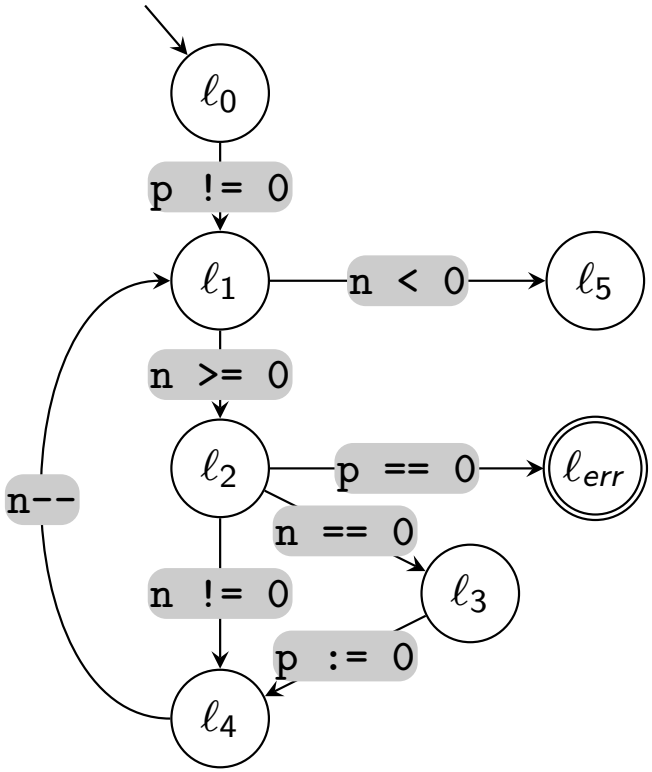
  Nodes: All explicit states
  Edges: "Program allows transition from source state to target state"

Traces, Executions

- *Trace* of program P:
  sequence of statements that respects control flow
- A trace is *infeasible* if there is no execution that corresponds to it.
- *Execution* of P:
  sequence of states and statements *that is possible* in P

Traces, Executions – Examples

```
ℓ0:    assume p != 0;
ℓ1:    while(n >= 0)
       {
ℓ2:        assert p != 0;
           if(n == 0)
           {
ℓ3:            p := 0;
           }
ℓ4:        n--;
       }
```



Abstraction – Example

To prove: $y \neq 0$ at any point in execution

Abstraction – Terminology

- Abstract state
- Abstract interpretation
  - Abstract domain
  - Abstract post
- Symbolic execution
- CEGAR – counter-example guided abstraction refinement

Logics-related Terminology

- SMT, SMT solver – "Satisfiability Modulo Theories"
  Theory (here): Set of Axioms
- Model checking, model checker
  $\varphi \overset{?}{\models} \psi$

Testing, Coverage

- “Falsification, not verification”
- Test case
  - Program input
  - Expected result
  - Environment information, “test harness”, ...
- Coverage – how “good” /complete is my set of tests?
  - Path coverage
  - Statement coverage
  - etc.

Miscelleanous terms

- Dynamic/static [analysis/...]
  dynamic: through executing the program
  static: without executing the program
- Symbolic/explicit [state/model checking/...]
- Safety/liveness properties
  safety: Can something bad happen?
  liveness: Will something good happen?

Part 2 – Reading a paper, giving a talk – Disclaimer

- Collection of hints/warnings
- Not a general guide to rhethorics/presentation
- Not a set of rules

The authors are not your friends (perhaps)

What the paper's authors want (among other things):

- Provide an answer to a specific question (you have never heard about)
- Get the paper accepted in peer review
  - Emphasiye that the contents are particularly new, hard to find, useful, ...
  - Hide (or at least not emphasize) weaknesses of the approach

Reading the paper – procedure

1. First overview: Read abstract, introduction, conclusion; perhaps look up unknown terminology, concepts
2. Second overview: Read paper as a whole, possibly skip intricate, technical parts;
3. Make a selection on what is important, what to present
4. On the important parts: Go into detail, try to understand every line

Hints

- Ask questions
- Try to read "between the lines"

Your talk

Hints

- Rhethorics:
  do everything that helps to get your content across
  avoid distractions
- What you show should make sense *within the talk*
- What you show (on your slides) $\neq$ what you say – think about both!
- Find the right speed. One slide one idea.

Formal recommendations

- Page numbers
- If using LaTeX:

  `\beamertemplatenavigationsymbolsempty`

Further Reading

- Very general advice on beamer presentations:
  Death by PowerPoint by Alexei Kapterev
  `http://www.slideshare.net/thecroaker/death-by-powerpoint`
- Background on how research papers work, some advice is also good for your talk.
  How to write a bad research paper? by Rachid Guerraoui
  `https://www.youtube.com/watch?v=K9BhQaOdtjs`