Verificação Formal de *Software*

Maria João Frade e Jorge Sousa Pinto
`{mjf,jsp}@di.uminho.pt`

O módulo num slide

- Verificação *Dedutiva* de Programas
- 12 sessões: 6 JSP + 6MJF
- Noções fundamentais de lógica de 1a. ordem do ponto de vista da verificação: limitações da prova automática; prova interactiva; ferramentas de prova [MJF]
- Semântica axiomática de programas; condições de verificação; ferramentas de verificação; outros tópicos [JSP]

Avaliação

- 3 componentes de avaliação:

  0.3*P1 + 0.3*P2 + 0.4*T

- P1 = estudo autónomo e apresentação de uma ferramenta de verificação (grupos de 2)
- P2 = pequeno trabalho de verificação em Frama-c (grupos de 2)
- T = Teste final

Ferramentas para PI

software model checkers & static analysers

- SLAM
- BLAST
- Coverity Prevent
- (East London Massive) Space Invader
- Goanna
- Splint
- CodeSurfer

Algumas Questões para Discussão Prévia

- O que se entende por Verificação Formal?
- E por Verificação de Programas?
- O que é um *model-checker*?

Formal Methods and Program Verification

The Central Problem of FM

Part 1: *model validation*

- How to enforce, at the specification level, the desired behaviour?

  Prove properties about the model

Tools for Formal Verification

- Proof Systems:

  Theorem Provers / Proof Assistants

- Model Checkers

The Central Problem of FM

Part 2: *relation between specifications and implementations*

- How to obtain, from a specification, an implementation with the specified behaviour? *Extraction; Program Derivation*

  Or alternatively,

- Given an implementation, how can it be checked that it obeys the specification? *Testing; Program Verification*

Program Extraction

- From a proof of a logical property (typically concerning existential quantifications), the Coq system is capable of extracting a program into a working programming language

Program Derivation

- Stepwise Refinement from Specifications to Programs

  (Z, VDM, B, …)

- Two approaches to correctness:

  (i) the refinement steps generate *proof obligations* that must be discharged. Derivations are thus formally verified.

  (ii) the refinement process is itself verified to be correct. The derived programs are then *correct by construction*.

Program Verification

- Given a program and a specification, check that the former conforms to the latter.
- This is the only applicable method in many situations
- approach in this course: based on *program annotations* and Hoare Logic

Program Verification

- One Possible Definition:

  “an exhaustive, correct and complete form of static checking w.r.t. to a specification”

- Properties may include functional aspects; safety properties; security properties; …

- Provides a global certification that the program behaves as it is specified to behave