Verifying Z Specifications with Z/Eves

AI4FM kick-off meeting talk

Andrius Velykis

PhD student at Newcastle University, AI4FM project
Supervised by Prof Cliff Jones

School of Computing Science
Newcastle University
20th May 2010

Overview

- Mechanising Z specifications
- Z/Eves theorem prover
- Verification activities
- Outcomes & conclusions

Mechanising Z specifications

- Grand Challenge (GC) in Verified Software
  - Pilot projects: Mondex, POSIX file-store, verified OS kernels
- Existing Z specifications
- Manual or no formal proofs
- Mechanising the specification
  - Perform/verify proofs
  - Component reusability
  - Model extensibility

Z/Eves theorem prover (I)

- Successfully used for Z specification and mechanisation in GC pilot projects
- Z syntax & type checking
- User friendly/easy learning curve
  - Strict & limited number of proof tactics
- EVES proof system with Z extensions
- Step-by-step and push-button proof approaches

Z/Eves theorem prover (II)

- Supports sizeable specifications
  - Mondex benchmarks:
    - ~200 paragraphs and >300 proofs
    - >4500 proof commands
    - ~1 hour to complete proofs (laptop, 2006)

- *No active development & distribution*

Z notation

- Based on Zermelo-Fraenkel set theory and first-order predicate logic
- Z schema
  - Named record with invariants
  - Can be included in other schemas
- State & operations specification style
  - Operation schema – before and after state with transition invariants
  - Supported by Z/Eves

Verification with Z/Eves (I)

- Syntax & type checking automatic
- Domain checking
  - Z notation does not impose undefinedness treatment
  - Well-defined partial functions (Z/Eves)
  - Generates domain check proof obligations (PO) automatically
  - Conditional expressions (J. McCarthy):

$$a \land b \mathrel{\widehat=} \textbf{if } a \textbf{ then } b \textbf{ else } false$$

Verification with Z/Eves (II)

- Inconsistency checking
  - Checks the feasibility of the specified model
  - Axiomatic definitions

$$\exists\, AxiomVariable \bullet AxiomPredicate$$

  - Initial state

$$\exists\, State' \bullet StateInit$$

Verification with Z/Eves (III)

- Precondition calculation
  - Determines the state under which operation is successful
  - Feasibility of operation
  - Verify API robustness

$$\text{pre } Operation = \exists State' \bullet Operation \setminus outputs$$

$$\forall State \bullet \text{pre } Operation$$

- Verified refactoring
- Properties of interest

Verification with Z/Eves (IV)

- No proof obligation (PO) generation
  - Can be extended – Z/Eves uses Z/*LaTeX*
- Choose the amount of push-button proof
  - Need *housekeeping* rules to support high levels of automation
  - Extracted generic rules & theorems
  - Some theorems not-automatable – need manual declaration of proof commands

Mechanisation outcomes

- Finding missing state and precondition invariants
- Model extension with security, functional and other properties
- Reusable verified data structures
- Generic toolkit lemmas
- Refinement & code generation
- Case studies, benchmarks, verification exercises

Summary

- Tool support for mechanisation of existing Z specifications
- Z/Eves – a user friendly theorem prover for Z specifications
- Increasing repository of case studies, proofs and reusable lemmas

- Questions?