Tool Support
for
Formal Proof Development

Roger B. Jones
International Computers Limited,
Eskdale Road,
Winnersh,
Wokingham,
Berks RG11 5TT

tel: 0734 693131 x6536,
fax: 0734 697636
email: R.B.Jones@win0109.uucp
Topics

the role of formal methods in the development of critical systems

the role of proof in such developments

the requirement for tools to support proof related activities

the characteristics and capabilities of ProofPower
**PROOF themes**

- focus formality
- automate proof

**METHODs** supporting selective application of formality to best effect in combination with other methods.

**TOOLS** proving least cost proof development support integrated with support for other methods.
ILLUSTRATIVE METHODS

“STRUCTURED”

FORMAL with REFINEMENT

FORMAL for CRITICAL COMPONENTS

FORMAL for CRITICAL REQUIREMENTS
“STRUCTURED” DEVELOPMENT

REQUIREMENTS

FUNCTIONAL SPECIFICATION

DESIGN

IMPLEMENTATION

VALIDATION
STRUCTURED DEVELOPMENT - PROs and CONs

PRO

provides framework for systematic design against requirements prior to implementation

permits checking of design against requirements prior to implementation

CON

reliance on english language prior to implementation

validation cannot test all possible cases
FORMAL DEVELOPMENT with
REFINEMENT

REQUIREMENTS

FORMAL FUNCTIONAL SPECIFICATION

VERIFIED REFINEMENT TO CODE

VALIDATION
FORMAL DEVELOPMENT with REFINEMENT - PROs and CONs

PRO

greater precision in functional specification
permits proof in refinement process
verification covers all cases

CON

specification may be large
specification may be wrong
“design” process radically changed
lack of focus
FORMAL DEVELOPMENT for CRITICAL COMPONENTS

REQUIREMENTS supplemented by HAZARD ANALYSIS

IDENTIFY CRITICAL COMPONENTS

Use FORMAL approach for CRITICAL COMPONENTS

Use STRUCTURED approach for NON-CRITICAL COMPONENTS
FORMAL DEVELOPMENT for CRITICAL COMPONENTS - PROs and CONs

PRO

formal treatment focused on critical components

CON

most of previously mentioned problems

critical components may be incorrectly identified
FORMAL DEVELOPMENT for CRITICAL REQUIREMENTS

formalise critical requirements on SYSTEM

formally model architecture

formalise critical requirements on SUBSYSTEMS

verify architecture

repeat through structured design process

implement and verify using pre/post conditions
FORMAL DEVELOPMENT of CRITICAL REQUIREMENTS - PROs and CONs

PRO

formal treatment focused on critical requirements

identification of critical components formally verified

requirements on critical components formally verified

CON

lack of literature on techniques
PROCESSING of FORMAL SPECIFICATIONS

SYNTAX CHECKING

TYPE CHECKING

..................

CONSISTENCY PROOFS

SEMANTIC WELL-FORMEDNESS PROOFS

PRE-CONDITION SIMPLIFICATION

REFINEMENT VERIFICATION

..................

PROOF of CRITICAL PROPERTIES

CODE/HARDWARE VERIFICATION
REQUIREMENTS for PROOF TOOLS

SOUNDNESS/INTEGRITY

PRODUCTIVITY

ADAPTABILITY/EXTENDIBILITY
Productivity Factors

The following factors can each influence by an order of magnitude or more the cost effectiveness of formal analysis:

• Architectural Considerations.

  The top-level structuring of high level specifications and the high level structuring of the system itself can have a major impact on the costs of verifying critical properties.

• The Proof Tool

  The level of automation, and the extent and relevance of libraries in the proof tool has a major impact on proof development productivity.

• Staff Skills

  Variation in productivity of staff can be very substantial.
ProofPower Software and Services

Software Support for Formal Specification and Formal Reasoning

Consultancy and Training in Languages, Methods and Tools

Formal Analysis Subcontract

Collaborative Bids
Languages Supported by ProofPower

• NOW:
  – Standard ML (as metalanguage)
  – Higher Order Logic

• SOON:
  – Z
  – SAL (SPARK Annotation Language)

• EVENTUALLY (we hope):
  – ISO Standard Z
  – others
ProofPower Functionality

• Document Preparation/Printing:
  – using LaTeX “literate scripts” with extended fonts for document sources
  – indexes, cross reference and theory listings

• Syntax Check/Type Check (interactive or batch)

• Formal Reasoning (interactive or batch)

• Theory Management:
  – specifications and theorems held in theory hierarchy
  – programmable access to theory hierarchy
ProofPower

- CRITICAL REQUIREMENT

all theorems are true
if
all extensions are conservative

ARCHITECTURE

code for management of theory database and checking of proofs separated out and protected using abstract data type

code for all other functions (e.g. syntax checking, type inference, proof heuristics) written in SML, but non-critical.

System user extensible.