Presentation to Oxford PRG

Roger Jones

ICL Defence Systems

THE LCF PARADIGM

implement proof checker using
a TYPED FUNCTIONAL programming LANGUAGE
as META-LANGUAGE (e.g. SML)

abstract data type of THEOREMS
GUARANTEED SOUND by the type checker
(assuming the logic is well defined)

META-LANGUAGE is AVAILABLE TO the USER
for programming proofs and other customisation,
WITHOUT risk of COMPROMISING the CHECKER.

BENEFITS of the LCF PARADIGM

- HIGH ASSURANCE of SOUNDNESS
- EASY to CUSTOMISE and EXTEND
- COMPLETE USER CONTROL
  of PROOF STRATEGY
- RERUNNABLE PROOF SCRIPTS
- LEG WORK convertible to HEAD WORK
  by PROGRAMMING in META-LANGUAGE

on the FIDELITY of TRANSLATIONS (1)

SAFETY CRITERIA:

- PRESERVATION of ILL-TYPEDNESS
- PRESERVATION of INVALIDITY

Translations are safe in both respects, except for treatment of partiality, loose generics, tuples.

CRITERIA of CORRECTNESS:

- PRESERVATION of WELL-TYPEDNESS
- PRESERVATION of VALIDITY

This might be achievable with some adjustments to the semantics of Z! A new parser/printer would be necessary.

We believe (without detailed study) that our translations are mostly SAFE, but we have NOT attempted to achieve CORRECT translations.

FIDELITY of TRANSLATIONS (2)

EASILY ELIMINABLE PROBLEMS

- use of choice function for loose specifications

problems ELIMINABLE by NEW PARSER

- translation to primitive function space

possible MODIFICATIONS to Z SEMANTICS

- conservative extensions
- treatment of partial functions and equality
- loose generic definitions

POINTS of UNCERTAINTY

- constraints on given sets
- instantiation of given sets
- tuples
- schema operations