Automated Reasoning

Revision

Alan Bundy
Course contents

- **Logics**: propositional, first order, aspects of higher order logics and linear temporal logic
- **Formalized mathematics**
- **Interactive theorem proving**: introduction to theorem proving with Isabelle
- **Formal verification using** model checking
- **Proof planning and rippling**: AI approach used to automatically guide proofs e.g. inductive proofs
Natural Deduction in Propositional Logic

• A first look at propositional logic
  - Syntax
  - Semantics

• Natural deduction
  - introduction and elimination rules
  - proofs given as trees

• Introduction to Isabelle
Natural Deduction in Propositional Logic (II)

- Natural deduction in propositional logic
- Isabelle
  - methods: rule, erule, frule, drule, rule_tac etc
  - tactics: simp, auto
- Next time: introduction to first order logic
Natural Deduction in First-Order Logic

• **Introduction to FOL**
  - Syntax and Semantics
  - Substitution
  - Intro and elim rules for quantifiers

• **Isabelle**
  - Declaring predicates
  - Brief look at types
Representation

- **Representational Issues**
  - Axioms or definitions?
  - Predicates or functions?
  - Lists, multisets, and variadic functions
  - Higher Order Logic

- There are often many ways to build a theory: the developer's task is to find the best way

- **Soundness** is essential, **completeness** is nice
Unification Algorithms

- Algorithms for matching and unification
- Unification as equation solving
- Transformation rules for equation solving
- Building-in axioms
- Most general unifiers and classification
Rewrite Rules

- **Rewrite rules** are a powerful technique for automated reasoning.

- A rule set gives **canonical normal forms** if it is
  - (1) terminating; and
  - (2) locally confluent.

- We show (1) by finding a well-founded measure.

- We show (2) using critical pairs and the Knuth-Bendix procedure to try to make confluent set from a non-confluent set. (This does not always work.)

- In Isabelle, we have a lot of control over the rewrites.
LTL Model Checking

- Introduction to model checking
- Model systems using FSAs
- Model system behaviour using LTL
  - temporal operators $\circ$, $\Diamond$, $\Box$, $U$, $W$
- Next time: model checking using Spin
Model Checking with SPIN

Promela code

Correctness Claims

SPIN

pan.c
C-code of the verifier

C-compiler

pan
the verifier

Error trails

random simulation

interactive simulation

-error trail simulation

-error trails

-claim

-interactive simulation

-pan.c

-C-compiler
Model Checking with SPIN (II)

- Loops in Promela (do and goto)
- Verifying using never claims
- Generating never claims from LTL formulae
- Accept cycles
- Safety and liveness properties
Model Checking with SPIN (III)

• **Translation** of Promela to automata.

• Promela **semantics**.

• **Asynchronous products** of automata.
  - Interleaving concurrency

• **Automata expansion**
  - to distinguish states with different valuations.

• **Automata for never-claims**.

• **Search** of expanded, product automata.

• Advantages and disadvantages of model checking.
Formalised PIN Cracking

• API analysis exciting new area!
  – Used also in smartcards, POS devices, mobile phones, DRM, ...

• Some early successes, many problems remain
  – XOR constraints, probabilistic model checking look good
  – Many open theoretical questions
Induction

- **Recursive functions and datatypes**
- **Inductive proofs**
  - may need to generalise conjecture to be proven
  - may need to speculate and prove additional lemmas
Rippling: Heuristic Guidance for Inductive Proof (I)

- **Inductive proof** introduces new search problems.
  - But also new opportunities – have IH.
- **Move differences** to make IH match IC.
- **Proof plan** for induction based on rippling.
  - Describes common pattern of proof.
- **Rippling**: selective; bidirectional; terminating
Rippling: Heuristic Guidance for Inductive Proof (II)

- **Ripple analysis**: Induction rules chosen to suit rippling.
- Different **patterns** of proof breakdown suggest different **patches**.
- **Ripple breakdowns** suggest: **induction** revision; **lemma** speculation or **generalisation**.
- Implemented via **proof planning with critics**.