Formal Verification in an Industrial Context

Carsten Sinz

Symbolic Computation Group, WSI
University of Tübingen
Germany

Overview

- Industrial verification case studies:
  1. Logic-based configuration (DaimlerChrysler)
  2. Rule-based expert system (IBM)
- Implications for logical formalisms and provers
- Practical experiences

Case Study 1: Automotive Product Configuration

- Rules check and modify orders, generate parts-list:

  970 → 673 ∧ 260 all police cars (970) must be equipped with a high-capacity battery (673) and no model type indicator on boot (260)

  682 ← 513L ∨ 727L add equipment for fire extinguisher (682) if car goes to Belgium (513L) or Guatemala (727L)

  Z04 ∨ Z06 ⇢ P9476 add special sealing of driver's door (P9476) to parts-list if car is armored (of type Z04 and Z06)

- Up to approx. 1,500 variables and 10,000 rules
- Consistency of rule system? Implications of change?

  ⇒ **Propositional validation criteria, SAT-checker**

Case Study 2: Verification of IBM's System Automation

- Rule-based expert system controls and monitors large sets of applications
  (starting, stopping, error recovery, load balancing, dependencies)
- Rules (finite-domain logic, WHEN-THEN) compute action sequence to reach given goal state
- Verified subsystem: 74 variables, 41 rules
- No cycles in computed action sequences?

  ⇒ **Propositional verification criteria (via intermediate language ΔPDL), SAT-checker, BDDs**

Favorable Properties of Logical Formalism

- Support for finite domain variables
  - Groups of mutually exclusive variables very common in product configuration
  - Finite domain language already employed in IBM's rules

  ⇒ **Language of Boolean logic extended by selection operator $S_k(f_1,\ldots,f_n)$**
- Full formula structure
  - Conversion to CNF for large formula is time-consuming, increases formula size (or number of variables)

  ⇒ **No restriction to formulae in CNF**

Demands on Proof Procedure

- Support for extended propositional language
  - ⇒ **Selection operator incorporated into Davis-Putnam-style algorithm for full propositional logic (no CNF)**
- Explanation
  - Indispensable for both proofs and failed proof attempts
  - ⇒ **Proof explanation by generation of minimal unsatisfiable subformulae (MUS), counterexamples either by model generation (SAT) or BDDs**
  - Identification of generalized error patterns
  - ⇒ **Distinction between relevant and irrelevant variables, existential abstraction over irrelevant variables (BDDs)**

Practical Experiences

- Surprisingly fast proofs in configuration domain
  - All proofs (formulae with >1000 propositional variables) by state-of-the-art SAT checker in <1 sec!

  ⇒ **Possible reason: always small conflicting rule sets, thus existence of short resolution proofs that carry over to DP**
- User's demands should be taken seriously
  - Prefer notions of problem domain to logical terminology
  - Graphical user interface, ease of use
  - Customized checks, as specialized as possible
  - Good integration into work-flow

Summary

Two industrial case studies have shown similar results:

- Current SAT checking technology very powerful
- Adaptation of prover language and algorithms to industrial domains worthwhile
- Explanation of results (both positive and negative) indispensable

For more information see

http://www-sr.informatik.uni-tuebingen.de