Proof Automation in Kleene Algebra

Peter Höfner

Universität Augsburg

Joint work with Georg Struth (University of Sheffield)
October 12, 2007

Automated Deduction in Formal Methods

Observation: Formal methods are dominated by model checking and interactive theorem proving

Automated Deduction in Formal Methods

Observation: Formal methods are dominated by model checking and interactive theorem proving

Automated deduction:

- special purpose provers seem necessary
- difficult to design and implement

Automated Deduction in Formal Methods

Observation: Formal methods are dominated by model checking and interactive theorem proving

Automated deduction:

- special purpose provers seem necessary
- difficult to design and implement

Question: How can we integrate verification techniques into automated deduction?

Automated Deduction in Formal Methods

New approach: off-the-shelf theorem provers and counterexample search with computational algebras

Idea:

- algebras provide first-order equational calculus
- this can be handled by resolution and paramodulation

Results:

- off-the-shelf theorem provers are an alternative
- no special purpose prover needed
- right domain model is needed
  variants of Kleene algebras yield good level of abstraction
- the verification is often done in two layers
- theorem provers should be able to handle simple arithmetics

- > 300 theorems proved
- applications in formal methods and computer mathematics

- most of the proofs fully automated from scratch
- some complex theorems needed lemmas (no surprise)

http://www.dcs.shef.ac.uk/∼georg/ka

The Setting

Theorem prover / Counterexample generator:

- Prover9 / Mace4
- software engineer's approach
  - *no* sophisticated encodings
  - *no* refined proof orderings
  - *no* hints or proof planning
  - *no* excessive running times

The Setting

Theorem prover / Counterexample generator:

- Prover9 / Mace4
- software engineer's approach
  - *no* sophisticated encodings
  - *no* refined proof orderings
  - *no* hints or proof planning
  - *no* excessive running times

Algebra:

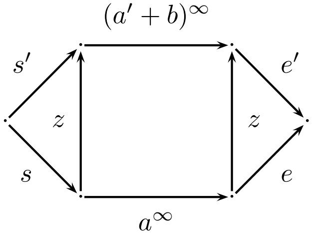
- Kleene algebras $(K, +, \cdot, 0, 1, ^*)$ (and variants)
  - elements are actions
  - $+$ models choice
  - $\cdot$ models sequential composition
  - * models finite iteration as a least fixedpoint
- rich model class: languages, relations, paths, traces, knowledge...

Case Studies

Refinement Calculus

A Classical Data Refinement Law [Back, vonWright]
Let $b^\infty = b^*$, $za' \leq az$, $zb \leq z$, $s' \leq sz$ and $ze' \leq e$. Then

$$s'(a'+b)^\infty e' \leq sa^\infty e.$$



$^\infty$ models finite arbitrary iteration (finite *or* infinite)

Results

- more complicated theorems also possible
  e.g., Back's atomicity refinement law

$$\frac{\begin{array}{c} s \le sq \qquad a \le qa \qquad qb = 0 \qquad rb \le br \\ (a+r+b)l \le l(a+r+b) \qquad q \le 1 \\ rq \le qr \qquad ql \le lq \qquad r^* = r^\infty \end{array}}{s(a+r+b+l)^\infty q \le s(ab^\infty q + r + l)^\infty}$$

- use proved lemmas
- sometimes restricted set of support
  - ping pong between Prover9 and Mace4
  - learning techniques (SRASS)
  - proved refinement laws instead of axioms

Hoare Logic

Exercise: Verify the following algorithm for integer division

funct $\mathrm{Div}(n, m)$
$k := 0$
$l := n$
while $m \leq l$ do
$k := k + 1$
$l := l - m$
return k

- precondition: $0 \leq n$
- postconditions: $n = km + l$, $0 \leq l$, $l < m$

Encoding in Hoare Logic: $\{p\}\ x_1\,;x_2\,;\text{while}\, r\, \text{do}\, y_1\,;y_2\, \text{od}\ \{q_1 \wedge q_2 \wedge \neg r\}$

Hoare Logic

Modal Kleene algebra

- Kleene algebra extended by *tests* and *modal operators* ($\langle x|p$, $|x\rangle p$, $[x|p$, $|x]p$)
- $\langle x|p$ is set of all states with at least one x-precessor in p

Encoding in Kleene algebra: $\langle x_1 x_2 (r y_1 y_2)^* \neg r | p \leq q_1 q_2 \neg r$

with

$$x_1 \hat{=} \{k := 0\}, \quad x_2 \hat{=} \{l := n\}, \quad y_1 \hat{=} \{k := k+1\}, \quad y_2 \hat{=} \{l := l - m\}, \quad r \hat{=} \{m \leq l\}$$

$$p \hat{=} \{0 \leq n\}, \quad q_1 \hat{=} \{n = km + l\}, \quad q_2 \hat{=} \{0 \leq l\}, \quad q_3 \hat{=} \{l < m\} = \neg r$$

Hoare Logic

Two-layered proof:

- *Step 1 (algebraic calculation)*
  - fully automated

$$p \leq |x_1]|x_2](q_1 q_2) \qquad \wedge \qquad q_1 q_2 r \leq |y_1]|y_2](q_1 q_2)$$
$$\Rightarrow \langle x_1 x_2 (r y_1 y_2)^* \neg r | p \leq q_1 q_2 \neg r$$

- *Step 2 (domain-specific reasoning)*
  - should be automated
  - assignment rule: $p[e/x] \leq |\{x := e\}]\, p$

$$\begin{aligned} |x_1]|x_2](q_1 q_2) &= |\{k := 0\}]\, |\{l := n\}](q_1 q_2) \\ &\geq (\{n = km + l\}\{0 \leq l\})[k/0][l/n] \\ &= \{n = 0m + n\}\{0 \leq n\} \\ &= \{0 \leq n\} \\ &= p \end{aligned}$$

Results

- often two-layered proofs
- concrete calculations, e.g., simple arithmetics are needed
- arithmetics should be included in theorem provers (SPASS+T)

Further Applications

- Rewrite Systems:
  - example: Church-Rosser theorems
- Linear temporal logic:
  - axioms are theorems or domain-specific
  - temporal reasoning about infinite systems
- Dynamic logic: axioms are theorems of modal Kleene algebra
- Modal correspondence theory:
  - Löb's formula related to frame property
  - calculational reasoning about infinite behaviour
  - alternative to translational approach
- Program refinement:
  - experiments in other variants of Kleene algebra
  - some complex refinement laws for action systems verified
- Relational methods:
  - > 100 theorems in relation algebra verified
  - example: $zx \sqcap y \leq (z \sqcap yx^\circ)(x \sqcap z^\circ y)$

http://www.dcs.shef.ac.uk/~georg/ka

Ongoing Work / Conclusion

Ongoing Work

- Knowledge and Games
- Network Flows
- Verification of Protocols
- Verification of Hybrid Systems

Ongoing Work / Conclusion

Ongoing Work

- Knowledge and Games
- Network Flows
- Verification of Protocols
- Verification of Hybrid Systems

Conclusion

- off-the-shelf theorem provers with computational algebras works
- light-weight formal methods with heavy-weight automation

