Algebra for Program Correctness in Isabelle/HOL

A. Armstrong V. B. F. Gomes G. Struth

University of Sheffield

April 28, 2014

Motivation

- algebras for program analysis and correctness
  - Kleene algebra with tests[1] (partial correctness)
  - demonic refinement algebra[2] (total correctness)
- reference formalisation in Isabelle/HOL[3]
- verification of program equivalences
- program construction and refinement
- verification with Hoare logic

Isabelle offers unique balance of expressivity/automation

---

[1] D. Kozen. Kleene algebra with tests. *ACM TOCL*, 1997.

[2] J. von Wright. Towards a refinement algebra, *SCP*, 2004.

[3] `http://afp.sourceforge.net/entries/KAT_and_DRA.shtml`

Motivation

Benefits of algebra

- program analysis by simple equational reasoning
- fits well with automated theorem proving
- separation of concerns (control flow *vs* data flow)
- data flow can be analysed by other means

How can algebras be integrated into program analysis tools?

Contributions

- formalisation of demonic refinement algebra in Isabelle/HOL
- 3 different axiomatisations for Kleene algebra with tests
- large libraries for KAT and DRA
- proof of classical program transformation examples
  - Back's atomicity refinement theorem
  - Kozen's loop transformation theorem
- computational models for KAT/DRA
  - binary relations
  - conjunctive/disjunctive predicate transformers
- principled approach to verification/refinement tools
- tools are themselves correct by construction

Kleene Algebras

Definition

KA is a structure $(K, +, \cdot, ^*, 0, 1)$ where

- $(K, +, \cdot, 0, 1)$ is a idempotent semiring, or dioid,
- with order defined by $x \leq y \longleftrightarrow x + y = y$
- the following fixpoint axioms hold for *

$$
\begin{array}{ll}
1 + x^*x \leq x^* \qquad & z + yx \leq y \to zx^* \leq y \\
1 + xx^* \leq x^* \qquad & z + xy \leq y \to x^*z \leq y
\end{array}
$$

Kleene Algebras with Tests

Definition

KAT is a structure $(K, B, +, \cdot, ^*, ^-, 0, 1)$ where

- $(K, +, \cdot, ^*, 0, 1)$ is a KA
- $(B, +, \cdot, ^-, 0, 1)$ is a BA with $B \subseteq K$

Algebraic programs semantics

$$\textbf{if } p \textbf{ then } x \textbf{ else } y \textbf{ fi} = px + \overline{p}y$$
$$\textbf{while } p \textbf{ do } x \textbf{ od} = (px)^*\overline{p}$$

Theorem

binary relations form KATs

Demonic Refinement Algebras

Definition

DRA is a structure $(K, +, \cdot, ^*, ^\infty, 0, 1)$ where

- $(K, +, \cdot, ^*, 0, 1)$ is almost a KA
- $x0 = 0$ fails
- the following axioms hold for $^\infty$

$$1 + xx^\infty \leq x^\infty \qquad y \leq xy + z \rightarrow y \leq x^\infty z$$
$$x^\infty = x^* + x^\infty 0$$

Demonic Refinement Algebras

Possibly infinite loop

$$\textbf{while } p \textbf{ do } x \textbf{ od} = (px)^{\infty}\overline{p}$$

Theorem

conjunctive/disjunctive predicate transformers form DRAs

Notation

refinement community uses $\sqcap$, ;, $^{\omega}$, $\top$, $\bot$, and $\sqsubseteq$

Kozen's Loop Transformation Theorem

Theorem

every sequential while program, appropriately augmented with subprograms of the form $z(pq + \overline{pq})$, can be viewed as a while program with at most one loop under certain preservation assumptions [1,2].

---

[1] D. Kozen. Kleene algebra with tests. *ACM TOCL*, 1997.

[2] K. Solin. Normal forms in total correctness for while programs and action systems. *JLAP*, 2011.

Kozen's Loop Transformation Theorem

what do * and $^\infty$ have in common?

Definition

a *pre*-Conway algebra is a dioid (without $x0 = 0$) where

$$(x + y)^\dagger = (x^\dagger y)^\dagger x^\dagger$$
$$(xy)^\dagger = 1 + x(yx)^\dagger y$$
$$zx \le yz \to zx^\dagger \le y^\dagger z$$

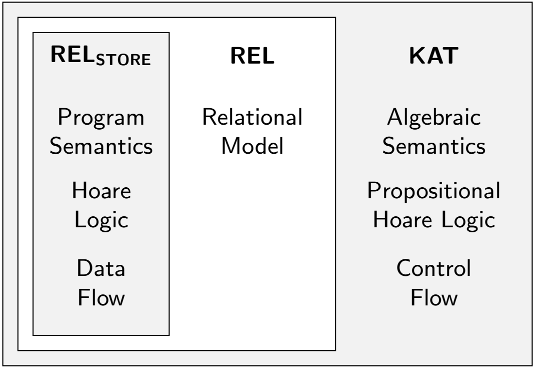
Remark

adding $1^\dagger = 1$ yields KA

Theorem

Kozen's transformation theorem holds in pre-Conway algebras (hence in KAT and DRA)

Verification Tool



principled approach based on algebra

Propositional Hoare Logic

Validity of Hoare triple

$$\vdash \{p\}\ x\ \{q\} \Leftrightarrow px\overline{q} = 0$$

Theorem

inference rules of propositional Hoare logic are theorems of KAT

$$\begin{aligned}
& & \vdash \{p\}\ \mathbf{skip}\ \{p\} \\
p \le p' \wedge q' \le q \wedge \vdash \{p'\}\ x\ \{q'\} & \;\Rightarrow & \vdash \{p\}\ x\ \{q\} \\
\vdash \{p\}\ x\ \{r\} \wedge \vdash \{r\}\ y\ \{q\} & \;\Rightarrow & \vdash \{p\}\ x; y\ \{q\} \\
\vdash \{pb\}\ x\ \{q\} \wedge \vdash \{p\overline{b}\}\ y\ \{q\} & \;\Rightarrow & \vdash \{p\}\ \mathbf{if}\ b\ \mathbf{then}\ x\ \mathbf{else}\ y\ \mathbf{fi}\ \{q\} \\
\vdash \{pb\}\ x\ \{p\} & \;\Rightarrow & \vdash \{p\}\ \mathbf{while}\ b\ \mathbf{do}\ x\ \mathbf{od}\ \{\overline{b}p\}
\end{aligned}$$

Store and Assignments

Store in Isabelle

- *store* S is implemented as record of program variables
- works for any type of data value
- each variable has a *retrieve* and an *update* function
- a state σ is an element of the store

Definition

Assignment statements are formalised as

$$(x := e) = \{(\sigma, x\_update\ \sigma\ e) \mid \sigma \in \mathsf{S}\}$$

Hoare Logic

Theorem
Hoare's assignment rule is derivable in relational KAT

$$P \subseteq Q[e/x] \;\Rightarrow\; \vdash \{P\}\;(x := e)\;\{Q\}$$

where $Q[e/x]$ denotes substitution of x by e in Q

Verification Tool

Control Flow

- Isabelle libraries for KAT include Hoare rules
- *hoare* tactic generates verification conditions
- these blast away control structure

Data Flow

- modelled in relational KAT
- integrates Isabelle libraries for data domains
- analysed with ATP systems and SMT solvers
- all proofs are internally reconstructed by Isabelle

Verification of Insertion Sort

```
lemma insertion_sort:
 `i := 1;
 while {| `i < |`A| |}
 do
   `j := `i;
   while {| 0 < `j ∧ `A ! `j < `A ! (`j-1) |}



   do
     `k := `A ! `j;
     `A ! `j := `A ! (`j-1);
     `A ! (`j-1) := `k;
     `j := `j-1
   od;
   `i := `i+1
 od
```

Verification of Insertion Sort

```
lemma insertion_sort: "⊢ {| |Ao| > 0 ∧ ‘A=Ao |}
 ‘i := 1;
 while {| ‘i < |‘A| |}
 do
   ‘j := ‘i;
   while {| 0 < ‘j ∧ ‘A ! ‘j < ‘A ! (‘j-1) |}



   do
     ‘k := ‘A ! ‘j;
     ‘A ! ‘j := ‘A ! (‘j-1);
     ‘A ! (‘j-1) := ‘k;
     ‘j := ‘j-1
   od;
   ‘i := ‘i+1
 od
 {| sorted ‘A ∧ ‘A ~π Ao |}"
```

Verification of Insertion Sort

```
lemma insertion_sort: "⊢ ⦃ |Ao| > 0 ∧ ‘A=Ao ⦄
 ‘i := 1;
 while ⦃ ‘i < |‘A| ⦄ inv ⦃ sorted (take ‘i ‘A) ∧ ‘A ∼π Ao ⦄
 do
   ‘j := ‘i;
   while ⦃ 0 < ‘j ∧ ‘A ! ‘j < ‘A ! (‘j-1) ⦄
   inv ⦃ (sorted_but (take (‘i+1) ‘A) ‘j) ∧ (‘i < |‘A|)
     ∧ (‘j ≤ ‘i) ∧ (‘j≠‘i ⟶ ‘A ! ‘j ≤ ‘A ! (‘j+1))
     ∧ (‘A ∼π Ao) ⦄
   do
     ‘k := ‘A ! ‘j;
     ‘A ! ‘j := ‘A ! (‘j-1);
     ‘A ! (‘j-1) := ‘k;
     ‘j := ‘j-1
   od;
   ‘i := ‘i+1
 od
 ⦃ sorted ‘A ∧ ‘A ∼π Ao ⦄"
```

Verification of Insertion Sort

```
apply (hoare, auto)
```

hoare tactic generates 8 subgoals

```
apply (metis One_nat_def take_sorted_butE_0)
apply (metis take_sorted_butE_n One_nat_def ...)
apply (metis One_nat_def Suc_eq_plus1 le_less_linear ...)
apply (unfold sorted_equals_nth_mono sorted_but_def)
apply (smt nth_list_update)
apply (metis One_nat_def perm.trans perm_swap_array)
apply (smt nth_list_update)
by (smt perm.trans perm_swap_array)
```

Morgan's Refinement Calculus

Specification Statement

one single axiom added to KAT

$$\vdash \{p\}\ x\ \{q\} \Leftrightarrow x \leq [p, q]$$

Theorem

Morgan's refinement laws become derivable

$$\begin{aligned}
p \leq q \Rightarrow [p, q] &\sqsubseteq \mathbf{skip} \\
p' \leq p \wedge q \leq q' \Rightarrow [p, q] &\sqsubseteq [p', q'] \\
[0, 1] &\sqsubseteq x \\
x &\sqsubseteq [1, 0] \\
[p, q] &\sqsubseteq [p, r]; [r, q] \\
[p, q] &\sqsubseteq \mathbf{if}\ \ b\ \mathbf{then}\ [pb, q]\ \mathbf{else}\ [\overline{b}p, q]\ \mathbf{fi} \\
[p, \overline{b}p] &\sqsubseteq \mathbf{while}\ b\ \mathbf{do}\ [bp, p]\ \mathbf{od}
\end{aligned}$$

Morgan's Refinement Calculus

Theorem

refinement laws for assignment are derivable in relational model

$$P \subseteq Q[e/x] \Rightarrow [P, Q] \sqsubseteq (x := e)$$
$$Q' \subseteq Q[e/x] \Rightarrow [P, Q] \sqsubseteq [P, Q']; (x := e)$$
$$P' \subseteq P[e/x] \Rightarrow [P, Q] \sqsubseteq (x := e); [P'; Q]$$

Refinement of Insertion Sort

$$\llbracket\ |Ao| > 0 \wedge \text{`}A{=}Ao,\ sorted\ \text{`}A \wedge \text{`}A \sim_\pi Ao\rrbracket$$

$$\sqsubseteq$$

$\text{`}i := 1;$
while $\{\!|\text{`}i < |\text{`}A||\!\}$ **do**
$\quad\llbracket\ sorted\ (take\ \text{`}i\ \text{`}A) \wedge \text{`}i < |\text{`}A| \wedge \text{`}A \sim_\pi Ao,$
$\quad\ sorted\ (take\ (\text{`}i{+}1)\ \text{`}A) \wedge (\text{`}i{+}1) \leq |\text{`}A| \wedge \text{`}A \sim_\pi Ao\rrbracket;$
$\quad\text{`}i := \text{`}i{+}1$
od
by *refinement*

Refinement of Insertion Sort

```
⊑
‘i := 1;
while {|‘i < |‘A||} do
  while {| ‘j≠0 ∧ ‘A ! ‘j < ‘A ! (‘j-1) |} do
    [[ ‘j ≤ ‘i ∧ sorted_but (take (‘i+1) ‘A) ‘j
      ∧ (‘j≠‘i ⟶ ‘A ! ‘j ≤ ‘A ! (‘j+1)) ∧ ‘A ∼π Ao
      ∧ (‘i+1) ≤ |‘A| ∧ ‘j≠0 ∧ ‘A ! ‘j < ‘A ! (‘j-1),
      ‘j-1 ≤ ‘i ∧ sorted_but (take (‘i+1) ‘A) (‘j-1)
      ∧ (‘j-1≠‘i ⟶ ‘A ! (‘j-1) ≤ ‘A ! ‘j) ∧ ‘j≠0
      ∧ (‘i+1) ≤ |‘A| ∧ ‘A ∼π Ao]];
    ‘j := ‘j-1
  od;
  ‘i := ‘i+1
od
```

Refinement of Insertion Sort

$$\llbracket\ `j \le `i \wedge sorted\_but\ (take\ (`i+1)\ `A)\ `j$$
$$\wedge\ (`j \neq `i \longrightarrow `A\ !\ `j \le `A\ !\ (`j+1)) \wedge `A \sim_\pi Ao$$
$$\wedge\ (`i+1) \le |`A| \wedge `j \neq 0 \wedge `A\ !\ `j < `A\ !\ (`j-1),$$
$$`j-1 \le `i \wedge sorted\_but\ (take\ (`i+1)\ `A)\ (`j-1)$$
$$\wedge\ (`j-1 \neq `i \longrightarrow `A\ !\ (`j-1) \le `A\ !\ `j) \wedge `j \neq 0$$
$$\wedge\ (`i+1) \le |`A| \wedge `A \sim_\pi Ao\rrbracket$$

$$\sqsubseteq$$

```
‘k := ‘A ! ‘j;
‘A ! ‘j := ‘A ! (‘j-1);
‘A ! (‘j-1) := ‘k
```

Refinement of Insertion Sort

```
⟦ |Ao| > 0 ∧ ‘A=Ao, sorted ‘A ∧ ‘A ∼π Ao⟧

⊑

‘i := 1;
while ⦃‘i < |‘A|⦄ do
  while ⦃ ‘j≠0 ∧ ‘A ! ‘j < ‘A ! (‘j-1) ⦄ do
    ‘k := ‘A ! ‘j;
    ‘A ! ‘j := ‘A ! (‘j-1);
    ‘A ! (‘j-1) := ‘k;
    ‘j := ‘j-1
  od;
  ‘i := ‘i+1
od
```

termination remains to be shown . . .

Conclusion

Work so far

- formalisation of KAT and DRA in Isabelle/HOL
- basis for program verification and correctness
- reference formalisation with large libraries (50 pages A4)
- integration into simple verification and refinement tools
- full Isabelle code is available online[1]

please ask me for a demo

[1] Armstrong, Gomes, Struth. Kleene algebras with tests and demonic refinement algebras. *AFP*, 2014. http://afp.sourceforge.net/entries/KAT_and_DRA.shtml

Conclusion

Related work in Isabelle/HOL

- verification with Hoare logic[1]
- flowchart equivalence proofs and Hoare logic in SKAT[2][3]
- rely/guarantee based concurrency verification[4]

Extensions

- wlp based reasoning with modal KA
- total correctness with DRA and predicate transformers
- concurrency verification with CKA

---

[1] Nipkow. Winskel is (almost) right: towards a mechanized semantics textbook. *FSTTCS*, 1996.

[2] Angus, Kozen. Kleene algebra with tests and program schematology. 2001.

[3] Armstrong, Struth, Weber. Program analysis and verification based on KA in Isabelle/HOL, *ITP*, 2013.

[4] Armstrong, Gomes, Struth. Algebraic principles for RG style concurrency verification tools. *FM*, 2014.

Verification in RA

- reference formalisation of RA in Isabelle integrates KA
- integrating KAT requires interpreting tests
- this suffices for verification/refinement with RA
- for heterogeneous relations better use Coq

KAT vs SKAT

- SKAT is KAT plus assignment axioms
- these have been formalised in Coq and Isabelle
- we can derive assignment axioms in relational KAT
- verification with SKAT in Isabelle seems more tedious

Algebras in the Archive

- already there:
  variants of KA, KAT, DRA, RA, other regular algebras
- in the near future:
  modal KA, CKA, quantales and fixpoint laws

please contribute ...