COMP3610/6361
Principles of Programming Languages

Peter Höfner

Oct 17, 2023

Section 26

Add-On
Program Algebras:
Floyd-Hoare Logic meets Regular Expressions

Motivation

- CCS and other process algebra yield algebraic expressions, e.g.

$$a.b.\textbf{nil} + c.\textbf{nil}$$

- they also give rise to algebraic (semantic) equalities, e.g.

$$a.\textbf{nil} + a.\textbf{nil} = a.\textbf{nil}$$

- but how does algebra relate to Hoare triples

Beyond Floyd-Hoare Logic

some ‘optimisations’ are not possible within Floyd-Hoare logic

$$\frac{\{P\}\ \textbf{if}\ b\ \textbf{then}\ c\ \textbf{else}\ c\ \{Q\}}{\{P\}\ c\ \{Q\}}$$

(trivially) unprovable in Floyd-Hoare logic

Trace Model – Intuition

a program can be interpreted as set of program runs/traces

sets of traces $s_0 c_1 s_1 c_2 \dots s_{n-1} c_{n_1} s_n$

$$A \subseteq \Sigma \times (Act \times \Sigma)^*$$

non-deterministic choice	$A \cup B$
sequential composition	$AB = \{asb \mid xs \in A \wedge sb \in B\}$
iteration	$A^* = \bigcup_{n \geq 0} = A^0 \cup A^1 \cup A^2 \dots$
skip	$1 = \Sigma$ (all traces of length 0)
fail/abort	$0 = \emptyset$

Guarded Commands – Intuition

a program can be interpreted as set of guarded commands

sets of guarded strings $\alpha_0 c_1 \alpha_1 c_2 \ldots \alpha_{n-1} c_{n_1} \alpha_n$
($\alpha, \beta, \ldots$ Boolean expressions)

non-deterministic choice	$A \cup B$
sequential composition	$AB = \{a\alpha b \mid x\alpha \in A \wedge \alpha b \in B\}$
iteration	$A^* = \bigcup_{n \geq 0} = A^0 \cup A^1 \cup A^2 \ldots$
skip	$1 = \{$all Boolean expressions$\}$
fail/abort	$0 = \emptyset$

Properties

- associativity: $a(bc) = (ab)c$
- neutrality: $1a = a = a1$
- distributivity: $(a + b)c = ac + bc$
  $a(b + c) = ab + ac$ (?)
- absorption: $0a = 0 = a0$
- iteration: $(ab)^* a = a(ba)^*$

Regular expressions

we know these rules from regular expressions, finite automata and formal languages

Kleene Algebra (KA)

is the algebra of *regular expressions*
(traces/guarded commands without ‘states’)

Examples

- $ab + ba$
  $\{ab, ba\}$

- $(ab)^*a = a(ba)^*$
  $\{a, aba, ababa, \dots\}$

- $(a + b)^* = (a^*b)^*a^*$
  $\{$all strings over a,$b\}$

Regular Sets – Intuition

regular sets over Σ

non-deterministic choice ($+$, $\mid$)	$A \cup B$
sequential composition	$AB = \{ab \mid x \in A \wedge b \in B\}$
iteration	$A^* = \bigcup_{n \geq 0} = A^0 \cup A^1 \cup A^2 \ldots$
neutral	$1 = \{\varepsilon\}$
	(language containing the empty word)
empty language	$0 = \emptyset$

Axioms of Kleene Algebra

A *Kleene algebra* is a structure $(K, +, \cdot, 0, 1, ^*)$ such that

- K is an *idempotent semiring* under $+$, $\cdot$, 0, 1

$$\begin{array}{ll} (a+b)+c = a+(b+c) & (a, \cdot b) \cdot c = a \cdot (b \cdot c) \\ a+b = b+a & a \cdot 1 = 1 \cdot a = a \\ a+a = a & a \cdot 0 = 0 \cdot a = 0 \\ a+0 = a & \end{array}$$

$$a \cdot (b+c) = a \cdot b + a \cdot c$$
$$(a+b) \cdot c = a \cdot c + b \cdot c$$

- $a^*b =$ least x such that $b + ax \leq x$
- $ba^* =$ least x such that $b + xa \leq x$

$x \leq y \Leftrightarrow x + y = y$
multiplication symbol is omitted

Characterising Iteration

- complete semiring/quantales (suprema exist)

$$a^* = \Sigma_{n \geq 0}\, a^n$$

  supremum with respect to $\leq$

- Horn axiomatisation
  - $a^* b =$ least x such that $b + ax \leq x$:

$$1 + aa^* \leq a*$$
$$b + ax \leq x \Rightarrow a^* b \leq x$$

  - $ba^* =$ least x such that $b + xa \leq x$:

$$1 + a^* a \leq a*$$
$$b + ax \leq x \Rightarrow ba^* \leq x$$

Models & Properties

regular expressions, traces and guarded strings form Kleene algebras

abstract laws: $(ab)^*a \leq a(ba)^*$
(proof is a simple exercise)

applies to all models

guarded strings/commands have more structure (assertions)

Australian National University

Kleene Algebra with Tests (KAT)

A *Kleene algebra with tests* is a structure $(K, B, +, \cdot, ^*, \neg, 0, 1)$, such that

- $(K, +, \cdot, ^*, 0, 1)$ is a Kleene algebra
- $(B, +, \cdot, \neg, 0, 1)$ is a Boolean algebra
- $B \subseteq K$

- $a, b, c, \ldots$ range over K
- $p, q, r, \ldots$ range over B

Kleene Algebra with Tests (KAT)

$+$, $\cdot$, 0, 1 serve double duty

- applied to programs, denote choice, composition, fail, and skip, resp.
- applied to tests, denote disjunction, conjunction, falsity, and truth, resp.
- these usages do not conflict

$$pq = p \wedge q \qquad p + q = p \vee q$$

Models

- Trace models
  K: sets of traces $s_0 c_1 s_1 c_2 \ldots s_{n-1} c_{n_1} s_n$
  B: sets of traces of length 0
- Language-theoretic models K: sets of guarded strings
  $\alpha_0 c_1 \alpha_1 c_2 \ldots \alpha_{n-1} c_{n_1} \alpha_n$
  B: atoms of a finite free Boolean algebra

Modelling Programs

[Fischer & Ladner 79]

- $a \,;\, b = ab$
- **if** p **then** a **else** $c = pa + \neg pc$
- **while** p **do** $c = (pc)^* \neg p$

Floyd-Hoare Logic vs KAT

Theorem

KAT subsumes propositional Floyd-Hoare logic (PHL)
(Floyd-Hoare logic without assignment rule)

$\{p\}\ c\ \{q\}$ modeled by $pc = pcq$ (or $pc\neg q = 0$, or $pc\neg q \leq 0$)

Floyd-Hoare logic

$$\frac{\{p\}\ a\ \{q\} \quad \{q\}\ b\ \{r\}}{\{p\}\ ab\ \{r\}}$$

$$pa\neg q = 0 \wedge qb\neg r = 0 \Longrightarrow pab\neg r = 0$$

$$\frac{\{p \wedge r\}\ a\ \{q\} \quad \{p \wedge \neg r\}\ b\ \{q\}}{\{p\}\ \textbf{if}\ r\ \textbf{then}\ a\ \textbf{else}\ b\ \{q\}}$$

$$pra\neg q = 0 \wedge p\neg rb\neg q = 0 \Longrightarrow p(ra + \neg rb)\neg q = 0$$

$$\frac{\{p \wedge r\}\ a\ \{p\}}{\{p\}\ \textbf{while}\ r\ \textbf{do}\ a\ \{\neg r \wedge p\}}$$

$$pra\neg p = 0 \Longrightarrow p(ap)^*\neg(\neg rp) = 0$$

Crucial Theorems

Theorem

These are all theorems of KAT

(proof is an exercise)

Theorem (Completeness Theorem)

All valid rules of the form

$$\frac{\{p_1\}\ c_1\ \{q_1\} \quad \ldots \quad \{p_n\}\ c_n\ \{q_n\}}{\{p\}\ c\ \{q\}}$$

are derivable in KAT (not so in PDL)

Advantages of Kleene Algebra

- unifying approach
- equational reasoning + Horn clauses
  some decidability & automation
- but, missing out assignment rule of Floyd-Hoare logic

Other Applications of KA(T)

There are more applications

- automata and formal languages
  - regular expressions
- relational algebra
- program logic and verification
  - dynamic Logic
  - program analysis
  - optimisation
- design and analysis of algorithms
  - shortest paths
  - connectivity
- others
  - hybrid systems
  - . . .

Rely-Guarantee Reasoning

Hoare triple

$$\{p\}\ c\ \{q\} \ \Leftrightarrow\ pc\neg q = 0$$

But what about $\{P, R\}\ c\ \{G, Q\}$?

$$\begin{aligned}\{p, a_R\}\ c\ \{b_G, q\} \ &\Leftrightarrow \{p\}\ a_R \parallel c\ \{q\} \wedge c \leq b_G \\ &\Leftrightarrow p(a_R \parallel c)\neg q = 0 \wedge c \leq b_G\end{aligned}$$

needs algebra featuring parallel (we have seen one)

- $R \parallel (S + T) = R \parallel S + R \parallel T$
- $R \parallel (S \cdot T) = (R \parallel S) \cdot (R \parallel T)$
- $R \parallel (S \parallel T) = (R \parallel S) \parallel (R \parallel T)$

Rely-Guarantee Reasoning

Hoare triple

$$\{p\}\ c\ \{q\} \ \Leftrightarrow\ pc\neg q = 0$$

But what about $\{P, R\}\ c\ \{G, Q\}$?

$$\begin{aligned} \{p, a_R\}\ c\ \{b_G, q\} \ &\Leftrightarrow \{p\}\ a_R \parallel c\ \{q\} \wedge c \leq b_G \\ &\Leftrightarrow p(a_R \parallel c)\neg q = 0 \wedge c \leq b_G \end{aligned}$$

needs algebra featuring parallel (we have seen one)

- $R \parallel (S + T) = R \parallel S + R \parallel T$
- $R \parallel (S \cdot T) = (R \parallel S) \cdot (R \parallel T)$
- $R \parallel (S \parallel T) = (R \parallel S) \parallel (R \parallel T)$