# On the Algebraic Derivation of Garbage Collectors

Han-Hing Dang

Universität Augsburg
University

**13th Int'l Conference on Relational and Algebraic Methods in Computer Science**

September 18, 2012

# Motivation

*Considered Application*: **Garbage Collection**

## Characterise garbage formally ?

- view abstract memory states as (non-labelled) graphs $a$
- nodes represent objects, addresses, ...
- edges denote references, links between objects, ...
- specify a set of nodes $s$ as the *entry resources* of the system
- define all nodes unreachable from $s$ in $a$ as *garbage*

# Relations as a Concrete Calculus

- relations enable pointfree calculational reasoning
- represent arbitrary graphs as relations $R$
- reachability within $n$ edges:

$$(x, y) \in R^n \iff \exists \text{ path of length } n \text{ in } R \text{ from } x \text{ to } y$$

- Kleene star $^*$ for arbitrary finite powers
- represent sets of nodes as subidentity relations
- set of *reachable* nodes from $s$:

$$reach(s\,, R) =_{df} (s\,; R^*)^{\urcorner}$$

where ; denotes relational composition, $^{\urcorner}$ codomain/range

# An Abstract Formalisation

Capture general behaviour by the use **Modal Kleene algebras**

- more general than relation algebras
- axioms mainly expressible in first-order logic
  - amenable to fully automated reasoning

# Ingredients of a Modal Kleene Algebra

**Additive Monoid**

$$x + (y + z) = (x + y) + z \ ,$$
$$x + y = y + x \ ,$$
$$x + 0 = 0 \ , \ x + x = x \ ,$$
$$x \le y \ \Leftrightarrow_{df} \ x + y = y \ .$$

**Multiplicative Monoid**

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \ ,$$
$$x \cdot 1 = x \ , \quad 1 \cdot x = x \ .$$

**Distributivity and Annihilation**

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z) \ , \qquad (x + y) \cdot z = (x \cdot z) + (y \cdot z) \ ,$$
$$x \cdot 0 = 0 \ , \qquad 0 \cdot x = 0 \ .$$

# Ingredients of a Modal Kleene Algebra

**Tests and Complements**

$$p + \neg p = 1 \;, \qquad p \cdot \neg p = 0 = \neg p \cdot p \;.$$

**Codomain, Diamond and Box**

$$x \leq x \cdot x^{\urcorner} \;, \qquad (x \cdot p)^{\urcorner} \leq p \;,$$
$$\langle x |\, p =_{df} (p \cdot x)^{\urcorner} \;, \qquad [x |\, p =_{df} \neg \langle x |\, \neg p = \neg(\neg p \cdot x)^{\urcorner} \;.$$

**Kleene star, Iteration**

$$1 + x \cdot x^* \leq x^* \;, \quad x \cdot y + z \leq y \Rightarrow x^* \cdot z \leq y \;,$$
$$1 + x^* \cdot x \leq x^* \;, \quad y \cdot x + z \leq y \Rightarrow z \cdot x^* \leq y \;.$$

# Reachability in the Algebra

- abstract definition:

$$reach(p\,,a) \;=_{df}\; \langle a^*|\, p = (\, p \cdot a^* \,)^{\urcorner}$$

- some properties:

$$reach(0\,,a) = 0$$
$$reach(p+q\,,a) = reach(p\,,a) + reach(q\,,a)$$
$reach$ is isotone in both arguments
$$reach(reach(p\,,a)\,,a) = reach(p\,,a)$$
$reach(p\,,a)$ is the smallest fixpoint $\mu_f$ of $f(q) = p + \langle a|\, q$

# An Algebraic Derivation

- $\langle a|\, p$ represents all direct successors of $p$
- following properties are valid

$$reach(0\,,a) = 0 \qquad reach(p\,,a) = p + reach(\langle a|\, p\,,a)$$

  - left hand side : termination case
  - right hand side : recursive specification

# An Algebraic Derivation

- $\langle a|\, p$ represents all direct successors of $p$
- following properties are valid

$$reach(0\,,a) = 0 \qquad\qquad reach(p\,,a) = p + reach(\langle a|\, p\,,a)$$

  - left hand side : termination case
  - right hand side : recursive specification

Hence:

$$reach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 0$$
$$\text{else } p + reach(\langle a|\, p\,,a)$$

# An Algebraic Derivation

$$reach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 0$$
$$\text{else } p + reach(\langle a|\,p\,,a)$$

**Problem:** Algorithm does not terminate generally!

- Example: $p = \{(1,1)\}$ and $a = \{(1,2),(2,1)\}$

# An Algebraic Derivation

$$reach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 0 \\ \qquad\qquad\qquad \text{else } p + reach(\langle a|\,p\,,a)$$

**Problem:** Algorithm does not terminate generally!

- Example: $p = \{(1,1)\}$ and $a = \{(1,2),(2,1)\}$

**A solution:** $reach(p\,,a) = p + reach(\langle a|\,p\,,\neg p\cdot a)$

- as long as $p \neq 0$ holds and $a$ is finite, $\neg p \cdot a$ decreases

# An Algebraic Derivation

$$reach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 0$$
$$\qquad\qquad\qquad\quad \text{else } p + reach(\langle a|\,p\,,a)$$

**Problem:** Algorithm does not terminate generally!

- Example: $p = \{(1,1)\}$ and $a = \{(1,2),(2,1)\}$

**A solution:** $reach(p\,,a) = p + reach(\langle a|\,p\,,\neg p \cdot a)$

- as long as $p \neq 0$ holds and $a$ is finite, $\neg p \cdot a$ decreases

Hence:

$$reach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 0$$
$$\qquad\qquad\qquad\quad \text{else } p + reach(\langle a|\,p\,,\neg p \cdot a)$$

# Non-Reachability in the Algebra

- define

$$noreach(p\,,a) \;=_{df}\; \neg reach(p\,,a) = \neg\langle a^*|\,p = [a^*|\,\neg p$$

- dual properties

$$noreach(0\,,a) = 1$$
$$noreach(p+q\,,a) = noreach(p\,,a)\cdot noreach(q\,,a)$$
$$noreach \text{ is antitone in both arguments}$$
$$noreach(p\,,a) = noreach(reach(p\,,a)\,,a)$$
$$noreach(p\,,a) \text{ is the greatest fixpoint } \nu_f \text{ of } f(q) = \neg p\cdot[a|\,q$$

# A Dual Algorithm

By Boolean algebra and $noreach(p\,,a) = \neg reach(p\,,a)$ :

1. $reach(0\,,a) = 0 \quad \rightsquigarrow \quad noreach(0\,,a) = 1$
2. $reach(p\,,a) = p + reach(\langle a|\,p\,,\neg p \cdot a)$

   $\rightsquigarrow$

   $noreach(p\,,a) = \neg p \cdot noreach(\langle a|\,p\,,\neg p \cdot a)$

# A Dual Algorithm

By Boolean algebra and $noreach(p\,,a) = \neg reach(p\,,a)$ :

1. $reach(0\,,a) = 0 \quad \leadsto \quad noreach(0\,,a) = 1$
2. $reach(p\,,a) = p + reach(\langle a|\, p\,, \neg p \cdot a)$

   $\leadsto$

   $noreach(p\,,a) = \neg p \cdot noreach(\langle a|\, p\,, \neg p \cdot a)$

Hence:

$$noreach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 1 \\ \text{else } \neg p \cdot noreach(\langle a|\, p\,, \neg p \cdot a)$$

# Optimisations

**Tail recursion**

- use new argument $r$ to collect all $\neg p$ accumulations in

$$noreach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 1$$
$$\text{else } \neg p \cdot noreach(\langle a|\, p\,, \neg p \cdot a)$$

# Optimisations

**Tail recursion**

- use new argument $r$ to collect all $\neg p$ accumulations in

$$noreach(p\,,a) \quad = \quad \text{if } p = 0 \text{ then } 1 \\ \text{else } \neg p \cdot noreach(\langle a|\,p\,, \neg p \cdot a)$$

- initialise $r = 1$ at the beginning and define

$$tnoreach(r\,,p\,,a) \quad = \quad \text{if } p = 0 \text{ then } r \\ \text{else } tnoreach(\neg p \cdot r\,, \langle a|\,p\,, \neg p \cdot a)$$

# More Transformations

**Optimisations for an imperative form**

- $\neg p \cdot a$ deletes all $\neg p$ nodes and all incident edges in $a$

$$tnoreach(r\,,p\,,a) \quad = \quad \text{if } p = 0 \text{ then } r \\ \text{else } tnoreach(\neg p \cdot r\,, \langle a|\,p\,, \neg p \cdot a)$$

- deletion of reachable resources of course not desired
(may require a copy of $a$)

Guarantee termination **without** directly modifying $a$ ?

# More Transformations

$$tnoreach(r\,,p\,,a) \quad = \quad \text{if } p = 0 \text{ then } r$$
$$\text{else } tnoreach(\neg p \cdot r\,, \langle a|\,p\,, \neg p \cdot a)$$

- remember: initially $r = 1$ and $r - p \;=_{df}\; \neg p \cdot r$
- first argument decreasing while $p \neq 0$

# More Transformations

$$tnoreach(r\,, p\,, a) \quad = \quad \text{if } p = 0 \text{ then } r$$
$$\text{else } tnoreach(\neg p \cdot r\,, \langle a|\, p\,, \neg p \cdot a)$$

- remember: initially $r = 1$ and $r - p \;=_{df}\; \neg p \cdot r$
- first argument decreasing while $p \neq 0$

**Idea**:

- use value $\neg p \cdot r$ in second argument

# More Transformations

$$tnoreach(r\,,p\,,a) \quad = \quad \text{if } p = 0 \text{ then } r$$
$$\text{else } tnoreach(\neg p \cdot r\,, \langle a|\,p\,, \neg p \cdot a)$$

- remember: initially $r = 1$ and $r - p =_{df} \neg p \cdot r$
- first argument decreasing while $p \neq 0$

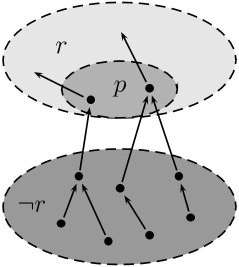
**Idea**:

- use value $\neg p \cdot r$ in second argument
- first transformation step:

$$tnoreach(r\,,p\,,a) = tnoreach(\neg p \cdot r\,, \neg p \cdot \langle a|\,p\,, \neg p \cdot a)$$

# More Transformations

To include $r$ assume invariant $\langle a|\, \neg r \leq p + \neg r$




and obtain

$$tnoreach(r\,, p\,, a) \quad = \quad \text{if } p = 0 \text{ then } r$$
$$\text{else } tnoreach(\neg p \cdot r\,, \neg p \cdot r \cdot \langle a|\, p\,, a)$$

# An Imperative Version

$$tnoreach(r\,,p\,,a) \quad = \quad \text{if } p = 0 \text{ then } r$$
$$\text{else } tnoreach(\neg p \cdot r\,, \neg p \cdot r \cdot \langle a|\, p\,,a)$$

- specification easily translated
- simple and short specification
- general and abstract form

noreach

```
p := roots; r := 1;
while (p != 0) {
   r := r − p;
   p := r · ⟨a|p;
}
return  r;
```

# CHARACTERISING CONCURRENT BEHAVIOUR

- consider a trace or sequence of elements $a_0, \ldots, a_n$
- specify and assume e.g. for all $0 \leq i \leq n$ the behaviour

$$s \leq p \Rightarrow \langle a_{i+1} | \, reach(p\,, a_i) \leq reach(p\,, a_i)$$

- direct successors of $reach(p\,, a_i)$ in $a_{i+1}$ are reachable in $a_i$
- allows simple and short algebraic proof of

$$s \leq p \Rightarrow noreach(p\,, a_i) \leq noreach(p\,, a_{i+1})$$

- garbage only grows or $reach(p\,, a_i)$ only decreases

# Outlook

**Further Ideas**

- concrete investigations on characterising concurrency
- representation of fragmented memory (compaction)
- further refinements of abstract algorithm
- calculate more complex derivational case studies
- use of sledgehammer tactic of *Isabelle* for (semi-)automated machine-checked proofs