Introduction to Separation Logic

Lectures at MGS'18

Georg Struth

on action short of strike at University of Sheffield, UK

Lecture 1: Statelets and Statelet Transformations

Plan

separation logic

- from algebraic point of view
- with some detours into algebra
- and Isabelle mathematical/verification components

lectures

1. statelets and statelet transformations
2. assertion algebra
3. predicate transformer semantics
4. verification conditions

exercises

depending on interest

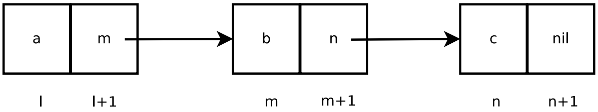
This Lecture

- brief introduction
- partial abelian monoids and heaplets
- partial abelian monoids and statelets
- faults and zeros
- statelet transformations

Linked List Reversal

list

$$[a, b, c]$$



program

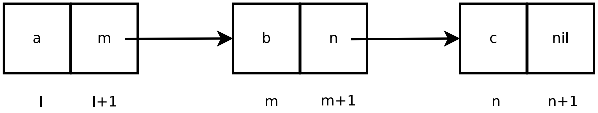
Y := nil;
while $\neg(X = \text{nil})$ do $Z := [X+1]; [X+1] := Y; Y := X; X := Z$ od

suppose X points to l

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$

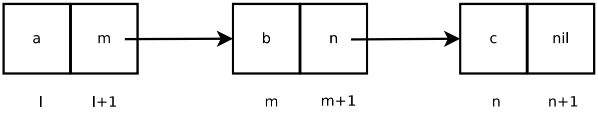


store	heap
$X = l,\ Y =?,\ Z =?$	$l \mapsto a,\ l+1 \mapsto m,\ m \mapsto b, m+1 \mapsto n,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$

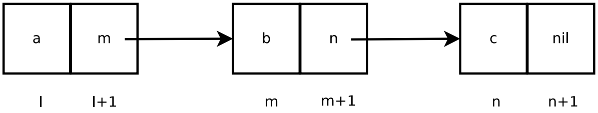


store	heap
$X = l,\ Y = nil,\ Z = ?$	$l \mapsto a,\ l+1 \mapsto m,\ m \mapsto b, m+1 \mapsto n,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X + 1]; [X + 1] := Y; Y := X; X := Z \text{ od}$

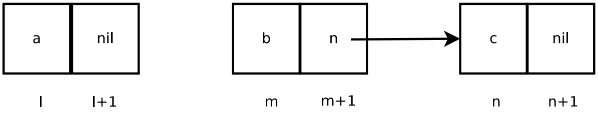


store	heap
$X = l,\ Y = nil,\ Z = m$	$l \mapsto a,\ l + 1 \mapsto m,\ m \mapsto b, m + 1 \mapsto n,\ n \mapsto c,\ n + 1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$

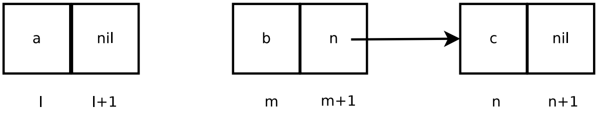


store	heap
$X = l,\ Y = nil,\ Z = m$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto n,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$

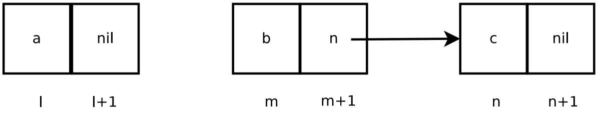


store	heap
$X = l,\ Y = l,\ Z = m$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto n,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$

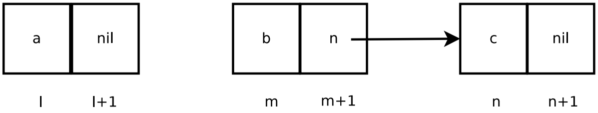


store	heap
$X = m,\ Y = l,\ Z = m$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto n,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$

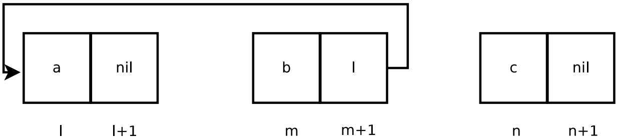


store	heap
$X = m,\ Y = l,\ Z = n$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto n,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X + 1]; [X + 1] := Y; Y := X; X := Z \text{ od}$

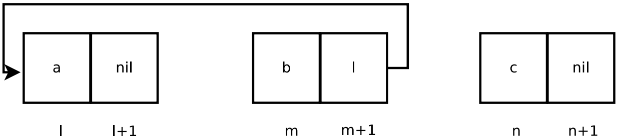


store	heap
$X = m,\ Y = l,\ Z = n$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto l,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y :=$ nil;

while $\neg(X =$ nil$)$ do $Z := [X+1]; [X+1] := Y; Y := X; X := Z$ od

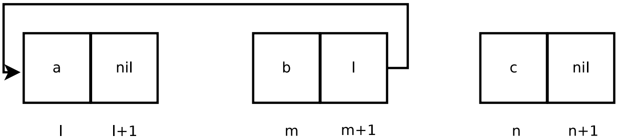


store	heap
$X = m,\ Y = m,\ Z = n$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto l,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y :=$ nil;

while $\neg(X =$ nil$)$ do $Z := [X+1]; [X+1] := Y; Y := X; X := Z$ od

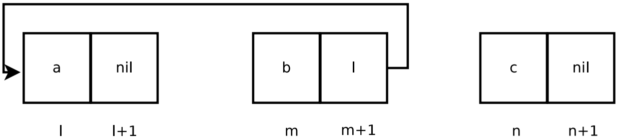


store	heap
$X = n,\ Y = m,\ Z = n$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto l,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X + 1]; [X + 1] := Y; Y := X; X := Z \text{ od}$

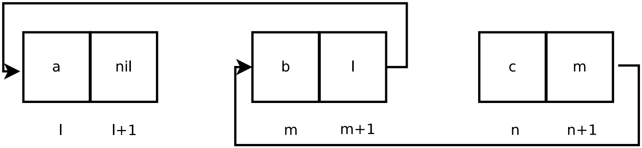


store	heap
$X = n,\ Y = m,\ Z = nil$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto l,\ n \mapsto c,\ n+1 \mapsto nil$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$

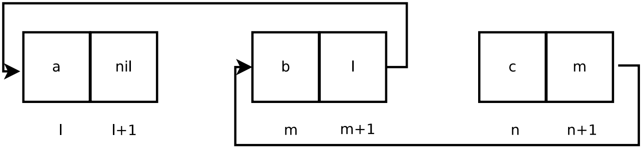


store	heap
$X = n,\ Y = m,\ Z = nil$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto l,\ n \mapsto c,\ n+1 \mapsto m$

Linked List Reversal

$Y := \mathrm{nil};$

$\mathrm{while}\ \neg(X = \mathrm{nil})\ \mathrm{do}\ Z := [X+1]; [X+1] := Y; Y := X; X := Z\ \mathrm{od}$

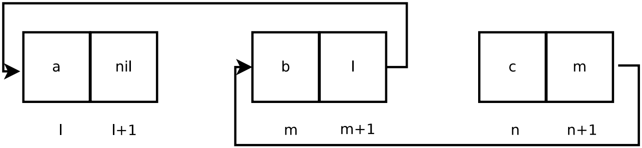


store	heap
$X = n,\ Y = n,\ Z = nil$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto l,\ n \mapsto c,\ n+1 \mapsto m$

Linked List Reversal

$Y := \text{nil};$

$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$



store	heap
$X = nil,\ Y = n,\ Z = nil$	$l \mapsto a,\ l+1 \mapsto nil,\ m \mapsto b, m+1 \mapsto l,\ n \mapsto c,\ n+1 \mapsto m$

Linked List Reversal

"Hoare triple"

$\{X$ points to linked list holding $\alpha\}$
$Y := \text{nil};$
$\text{while } \neg(X = \text{nil}) \text{ do } Z := [X+1]; [X+1] := Y; Y := X; X := Z \text{ od}$
$\{Y$ points to linked list holding $rev\,\alpha\}$

Defining Linked Lists

intuition

predicate list α e (σ, η) means that

- α is linked list in heap η
- starting at location specified by $e\ \sigma$ in store σ

definition

by recursion

$$\begin{aligned} \text{list } [\,]\ e &= (e \doteq nil) \\ \text{list } (x : xs)\ e &= \exists e'.\ e \mapsto x * e + 1 \mapsto e' * \text{list } xs\ e' \end{aligned}$$

remarks

- separating conjunction $*$ reads "and separately (in other heaplet)"
- with $\wedge$, absence of sharing not specified!

Linked List Reversal Formalised

Hoare triple

$\{\text{list}\,\alpha\,X\}$
$Y := \text{nil};$
$\text{while } \neg(X = \text{nil}) \text{ do } (Z := [X+1]; [X+1] := Y; Y := X; X := Z)$
$\{\text{list}\,(rev\,\alpha)\,Y\}$

invariant of while-loop

$$\exists \beta, \gamma.\ \text{list}\ \beta\ X * \text{list}\ \gamma\ Y \wedge (rev(\alpha) = rev(\beta) +\!\!+ \gamma)$$

separating conjunction captures absence of sharing between X and Y

extended Hoare logic needed to verify this program but let's start at the beginning. . .

Heaplets

- heap memory area as partial abelian monoid
- heaplets are pieces of a heap
- operations of heaplet addition/subtraction, subheaplet relation
- similar to resource monoids . . .
- . . . but well known from foundations of quantum mechanics

Partial Semigroups

partial semigroup

structure $(S, \cdot, D)$ with

- $D \subseteq S \times S$ domain of composition of $\cdot$
- $\cdot : D \to S$ partial operation
- for all $x, y, z \in S$

$$D\,x\,y \;\wedge\; D\,(x \cdot y)\,z \Leftrightarrow D\,y\,z \;\wedge D\,x\,(y \cdot z)$$
$$D\,x\,y \;\wedge\; D\,(x \cdot y)\,z \Rightarrow (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

intuition

- if either side of $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ is defined then so is other side
- and in this case both sides are equal

Partial Monoids

partial monoid

structure $(M, \cdot, D, E)$ with

- $(M, \cdot, D)$ partial semigroup
- $E \subseteq M$ such that

$$\exists e \in E.\ D\, e\, x \wedge e \cdot x = x \qquad \exists e \in E.\ D\, x\, e \wedge x \cdot e = x$$

$$e_1, e_2 \in E \wedge D\, e_1\, e_2 \Rightarrow e_1 = e_2$$

intuition

- every element has left/right unit
- in fact exactly one
- different units can't be composed

definition similar to Mac Lane's (meta)category axioms

Partial Abelian Monoids

partial abelian semigroup

partial semigroup $(S, \oplus, D)$ with

$$D\,x\,y \Rightarrow D\,y\,x \wedge x \oplus y = y \oplus x$$

partial abelian monoid (PAM)

partial monoid + partial abelian semigroup

Examples

monoids

every (abelian) monoid $(M, \cdot, 1)$ is a partial (abelian) monoid with $D = M \times M$ and $E = \{1\}$

ordered pairs

ordered pairs over X under cartesian fusion product $(a, b) \cdot (c, d) = (a, d)$ if $b = c$ form partial monoid with

$$D = \{((a, b), (c, d)) \mid b = c\} \qquad E = Id_X$$

Examples

intervals

- let $(X, \leq)$ be linear poset
- closed interval in X is ordered pair $[a, b]$ with $a \leq b$
- closed intervals under interval fusion $[a, b] \cdot [c, d] = [a, d]$ if $b = c$ form partial monoid on X^2 with D and E like for relations

segments

- let $(X, \leq)$ be a poset
- segment of X is is ordered pair (a, b) with $a \leq b$
- segments under segment fusion form partial monoid on X^2

Examples

paths

- let $G = (V, E)$ be (di)graph
- path in G is sequence $\pi = (v_1, \ldots v_n)$ of vertices along edges
- paths in G under path fusion (glueing ends together) form partial monoid on G with $E = V$

traces

- let $G = (V, E, \lambda)$ be edge-labelled (di)graph with $\lambda : E \to \Sigma$
- trace in G is sequence $\tau = (v_1, \sigma_1, \ldots, v_{n-1}, \sigma_{n-1}, v_n)$ along edges
- traces in G under trace fusion (glueing ends together) form partial monoid on G with $E = V$

Examples

multisets

multisets $f : X \to \mathbb{N}$ over X form (partial) abelian monoids under $(f + g)\,x = f\,x + g\,x$ and with $E = \{\lambda x.0\}$

sets

sets with $X + Y = X \cup Y$ if $X \cap Y = \emptyset$ form PAM with $E = \{\emptyset\}$

multisets are paradigmatic resources

Examples

(generalised) effect algebras

- let Hilbert space $\mathcal{H}$ represent some quantum system
- an effect over $\mathcal{H}$, a self-adjoint operator A on $\mathcal{H}$ such that $0 \leq A \leq id_{\mathcal{H}}$, represents unsharp measurement
- let $\mathcal{E}(\mathcal{H})$ be set of all effects over $\mathcal{H}$
- then $(\mathcal{E}(\mathcal{H}), \oplus, 0)$ with $A \oplus B = A + B$ if $A + B \leq id_H$ forms PAM

effect algebras have been studied for 25 years

Examples

heaplets

partial functions $X \rightharpoonup Y$ form PAM S_H with

$$\begin{aligned} \eta_1 \oplus \eta_2 &= \eta_1 \cup \eta_2 \\ D &= \{(\eta_1, \eta_2) \in S_H \times S_H \mid \mathit{dom}\,\eta_1 \cap \mathit{dom}\,\eta_2 = \emptyset\} \\ E &= \{\varepsilon\} \end{aligned}$$

where ε denotes empty heaplet

intuition

- heaplets are pieces of a heap
- $\oplus$ (heaplet addition) extends heaps by pieces
- it underlies heap allocation/mutation commands of separation logic

Remarks

- partial algebras have been studied for almost a century
- earliest reference I know is article by Brand (1927)
- PAMs are called resource monoids in separation logic

Remarks

- mutation/deallocation require more succinct description of heap
  1. heaplet subtraction operation
  2. subheap relation
- subtraction allows deleting pieces from heaps if these are subheaps

we study them abstractly in PAMs

Subheap Relation

Green's preorder

defined in every PAM as $x \preceq y \Leftrightarrow \exists z.\ D\,x\,z \wedge x \oplus z = y$

remark

$x \preceq y$ if and only if $x \oplus z = y$ (exists and) has solution in z

lemma

- $\preceq$ is precongruence: $x \preceq y \wedge D\,z\,x \Rightarrow z \oplus x \preceq z \oplus y$ (and $D\,z\,y$)
- every PAM is preordered by its Green's relation

Subheap Relation

- in the literature $\preceq = \preceq_R = \preceq_L$
- Green's relations R, L and H are associated congruences

Green's relations are the fundamental congruences of semigroup theory

Heaplet Subtraction

cancellation

PAM is cancellative if $D\,x\,z \wedge D\,y\,z \wedge x \oplus z = y \oplus z \Rightarrow x = y$

lemma

in cancellative PAM, if $x \preceq y$ then

- $x \oplus z = y$ is defined
- and has unique solution in z

subtraction

we write $y \ominus x$ for this solution

Heaplet Subtraction

lemma

in cancellative PAM

1. $D\,x\,z \wedge x \oplus z = y \Leftrightarrow x \preceq y \wedge z = y \ominus x$
2. $D\,x\,y \Rightarrow (x \oplus y) \ominus x = y$ and $x \preceq y \Rightarrow x \oplus (y \ominus x) = y$
3. if $x \preceq y$ then $D\,x\,z \wedge x \oplus z \preceq y \Leftrightarrow z \preceq y \ominus x$
4. $D\,x\,y \Rightarrow x \preceq x \oplus y$ and $x \preceq y \Rightarrow y \ominus x \preceq y$

Heaplet Subtraction

positivity

PAM is positive if $D\,x\,y \wedge x \oplus y \in E \Rightarrow x \in E$

lemma

Green's preorders are partial orders in positive cancellative PAMs

remark

positive cancellative PAMs with $E = \{1\}$ are known as generalised effect algebras in foundations of quantum mechanics

everything so far is known from foundations of physics

Heaplet Summary

in PAM S_H of heaplets

- $\eta_1 \preceq \eta_2$ iff η_1 is subheaplet of η_2
  - η_2 can be obtained by adding some piece to η_1
- S_H is cancellative and positive
  - adding different pieces to heaplet yields different heaplets
  - ε has no subheaplets
- $\preceq$ is partial order
- $\eta_1 \ominus \eta_2$ defined whenever η_2 is subheaplet of η_1
- $\oplus$ and $\ominus$ are inverses up-to definedness
- $\ominus$ needed for heap deallocation/mutation in separation logic

Statelets

- program states of separation logic are store-heap pairs
- they correspond to PAMs of cartesian products

Statelets

lemma

if X is a set and $(S, \oplus, D, E)$ a PAM

1. then $(X \times S, \oplus', D', E')$ forms PAM with

$$(x_1, y_1) \oplus' (x_2, y_2) = (x_1, y_1 \oplus y_2)$$
$$D' = \{((x_1, y_1), (x_2, y_2)) \mid x_1 = x_2 \wedge (y_1, y_2) \in D\}$$
$$E' = \{(x, e) \mid x \in X \wedge e \in E\}$$

2. if S is cancellative or positive, then so is $X \times S$

lemma

if X is a set and S a PAM then

1. $(x_1, y_1) \preceq (x_2, y_2) \Leftrightarrow x_1 = x_2 \wedge y_1 \preceq y_2$ is Green's order
2. $(x_1, y_1) \preceq (x_2, y_2) \Rightarrow (x_2, y_2) \ominus (x_1, y_2) = (x_1, y_2 \ominus y_1)$
   if $X \times S$ cancellative

Statelets

- heaplets have often type $L \rightharpoonup E$ with $L \subseteq E$
  - L is set of locations
  - E is set of expressions/values
  - locations/expressions are evaluated in store
- program store is set of functions of type $V \to E$
  - V is set of program variables
- store-heaplet pairs (σ, η) forms positive cancellative PAM S_S of statelets
  - substatelet relation $\preceq$ compares heaplets with same store
  - $\oplus$ and $\ominus$ on statelets adds/subtracts heaplets with same store
  - statelets have units $E_S = \{(\sigma, \varepsilon) \mid \sigma \in E^V\}$ . . . one per store

Faults and Zeros

- in program semantics, undefinedness is often captured in total setting by bottom elements
- in standard semantics of separation logic, these denote program faults due to partiality of heaplet operations

we now explain this relationship

Faults and Zeros

zeros

- annihilator 0 of PAM S satisfies $D\,0\,x$ and $0 \oplus x = 0$
- annihilators are unique whenever they exist

morphisms

- partial semigroup morphism $\varphi : S_1 \to S_2$ satisfies
  - $D_1\,x\,y \Rightarrow D_2\,(\varphi\,x)\,(\varphi\,y)$
  - $\varphi\,(x \oplus_1 y) = (\varphi\,x) \oplus_2 (\varphi\,y)$
- it is strong if $D_2\,(\varphi\,x)\,(\varphi\,y) \Rightarrow D_1\,x\,y$
- partial monoid morphism is partial semigroup morphism satisfying
  - $e \in E_1 \Rightarrow \varphi\,e \in E_2$
- it is strong if $\varphi\,e \in E_2 \Rightarrow e \in E_1$

Faults and Zerios

proposition

1. Every PAS (PAM with $E = \{1\}$) can be strongly embedded into an abelian semigroup (monoid) with zero
2. Every abelian semigroup (monoid with zero) contains a PAS (PAM with single unit) as submonoid

Faults and Zeros

example

- let $S_\perp = S \cup \{\perp\}$ for any PAM S
  - extend $\oplus$ to $\oplus_\perp$ such that $x \oplus_\perp y = \perp$ iff $(x, y) \notin D$
  - then $\perp \oplus_\perp x = \perp$ for any $x \in S_\perp$
  - extend $\preceq$ to $\preceq_\perp$
  - then $\perp \preceq_\perp x$ for all $x \in S$

  $(S_\perp, \oplus_\perp)$ forms an abelian semigroup (abelian monoid with unit 1 if $E = \{1\}$ in S)
- remove $\perp$ from abelian semigroup $S_\perp$
  - restrict $\oplus_\perp$ to $\oplus$ with $D = \{(x, y) \in S_\perp \times S_\perp \mid x \oplus_\perp y \neq \perp\}$

  $(S, \oplus, D)$ is PAS (PAM with $E = \{1\}$ if $S_\perp$ is abelian monoid with unit 1)

Faults and Zeros

example

- construction of semigroup (monoid) from $X \times S$ requires two zeros
  1. expand S to $S_{\perp_1}$ as before
  2. adjoin $\perp_2$ to the product PAS (PAM) which yields $(X \times S_{\perp_1})_{\perp_2}$
- the extensions of $\oplus$ and $\preceq$ follow the previous construction
- we write $\oplus_{\perp_2}$ and $\preceq_{\perp_2}$ at outer level
- this yields abelian semigroup
  - multiple units are forgotten in construction
  - $(x_1, x_2) \oplus_{\perp_2} (y_1, y_2) = \perp_2$ iff $x_1 \neq y_1$ or $x_2 \cdot y_2 = \perp_1$
  - then $(x_1, \perp_1) \oplus_{\perp_2} (y_1, y_2) = \perp_2$
- faults propagated from heaplets to statelets
- recovery of PAM $X \times S$ from $(X \times S_{\perp_1})_{\perp_2}$ straightforward
- instantiation to statelets $E^V \times S_H$ is straightforward as well

Statelet Dynamics

- $\oplus$ and $\ominus$ underly 3 of 5 basic commands of separation logic
  - heap mutation
  - heap allocation
  - heap deallocation
- heap lookup and store assignment are discussed as well
- we define state update function acting on PAM S_S for each of them
- if $s \in S_S$ is statelet then we write
  - $\sigma_s = \pi_1\, s$ for its store
  - $\eta_s = \pi_2\, s$ for its heaplet
- we use semi-algebraic approach in concrete model S_S

Addition/Subtraction of Single Heap Cells

domains of definition

$$D_{\oplus}\, s\,(\sigma_s, l \mapsto e) \Leftrightarrow l\,\sigma_s \notin dom\,\eta_s$$
$$D_{\ominus}\, s\,(\sigma_s, l \mapsto e) \Leftrightarrow l\,\sigma_s \in dom\,\eta_s \wedge e = \eta_s\,(l\,\sigma_s)$$

heap cell addition

update function $f_{\oplus} : E \to S_S \to \mathcal{P}\, S_S$ defined (nondeterministically) by

$$f_{\oplus}\, e\, s = \{(\sigma_s, \eta_s \oplus \{l\,\sigma_s \mapsto e\,\sigma_s\}) \mid l\,\sigma_s \notin dom\,\eta_s\}$$

heap cell deallocation

update function $f_{\ominus} : L \to S_S \to S_S$ defined by

$$f_{\ominus}\, l\, s = (\sigma_s, \eta_s \ominus \{l\,\sigma_s \mapsto \eta_s\,(l\,\sigma_s)\}) \qquad \text{if } l\,\sigma_s \in dom\,\eta_s$$

Heap Mutation

heap mutation

update function $f_m : L \rightarrow E \rightarrow S_S \rightarrow S_S$ defined by

$$f_m\, l\, e = (\hat{f}_\oplus\, l\, e) \circ (f_\ominus\, l)$$

where $\hat{f}_\oplus\, l\, e\, s = (\sigma_s, \eta_s \oplus \{l\,\sigma_s \rightarrow e\,\sigma_s\})$ if $l\,\sigma_s \notin dom\,\eta_s$

lemma

$$f_m\, l\, e\, s = (\sigma_s, \eta_s[l\,\sigma_s \leftarrow e\,\sigma_s]) \qquad \text{if } l\,\sigma_s \in dom\,\eta_s$$

where $f[x \leftarrow a]$ indicates that value of x in f has been updated to a

Store Assignment and Heap Lookup

store assignment

update function $f_a : V \rightarrow E \rightarrow S_S \rightarrow S_S$ defined by

$$f_a\, x\, e\, s = (\sigma_s[x \leftarrow e\, \sigma_s], \eta_s)$$

heap lookup

update function $f_l : V \rightarrow L \rightarrow S_S \rightarrow S_S$ defined by

$$f_l\, x\, l\, s = (\sigma_s[x \leftarrow \eta_s\, (l\, \sigma_s)], \eta_s) \qquad \text{if } e\, \sigma_s \in dom\, \eta_s$$

Heap Allocation

heap allocation

update function $f_c : V \to E \to S_S \to \mathcal{P}\, S_S$ defined by

$$f_c\, x\, e = (\mathcal{P}\,(f_a\, x)) \circ (f_\oplus\, e)$$

where $\mathcal{P}\, f$ computes image of given set under f

lemma

$$f_c\, x\, e\, s = \{(\sigma_s[x \to l\,\sigma_s], \eta_s \oplus \{l\,\sigma_s \mapsto e\,\sigma_s\}) \mid l\,\sigma_s \notin dom\,\eta_s\}$$

remark

- several cells are usually allocated in one fell-swoop
- such deterministic update functions can be obtained by refinement

Conclusion

- abstract PAM-based model of program states (statelets)
- link with algebraic fault model
- basic assignments of separation logic modelled by update functions that act on state space
  - store assignment
  - heap mutation
  - heap lookup
  - heap allocation
  - heap deallocation

next lecture: assertion algebra of separation logic

Exercises

?

Further Reading

- Calcagno et al, *Local Action and Abstract Separation Logic*
- Clifford, Preston, *The Algebraic Theory of Semigroups*
- Dongol, Gomes, Struth, *A Program Construction and Verification Tool for Separation Logic*
- Dongol, Hayes, Struth, *Convolution as a Unifying Concept*
- Foulis, Bennett, *Effect Algebras and Unsharp Quantum Logics*
- Gordon, *Lecture Notes on Hoare Logic*
- Hedlíková, Pulmannová, *Generalized Difference Posets and Orthoalgebras*
- O'Hearn, *A Primer on Separation Logic*
- Reynolds, *Separation Logic: A Logic for Shared Mutable Data Structures*
- Isabelle components: `https://www.isa-afp.org/entries/PSemigroupsConvolution.html`