AN INTRODUCTION TO

SEPARATION LOGIC

6. Iterated Separating Conjunction

John C. Reynolds
Carnegie Mellon University

February 16, 2011

—

Iterated Separating Conjunction

$$\langle \mathsf{assert} \rangle ::= \cdots \mid \bigodot_{\langle \mathsf{var} \rangle = \langle \mathsf{exp} \rangle}^{\langle \mathsf{exp} \rangle} \langle \mathsf{assert} \rangle$$

$$\bigodot_{v=e}^{e'} p \stackrel{\mathsf{def}}{=} (p/v \to e) \, * \, (p/v \to e+1) \, * \, \cdots \, * \, (p/v \to e').$$

More precisely,

$s, h \vDash \bigodot_{v=e}^{e'} p$ iff

$\quad \textbf{let } m = [\![e]\!]_{\mathsf{exp}} s, \; n = [\![e']\!]_{\mathsf{exp}} s, \; I = \{ \, i \mid m \leq i \leq n \, \} \textbf{ in}$

$\quad\quad \exists H \in I \to \mathsf{Heaps}.$

$\quad\quad\quad \forall i, j \in I. \; i \neq j$ implies $Hi \perp Hj$

$\quad\quad\quad$ and $h = \bigcup \{ \, Hi \mid i \in I \, \}$

$\quad\quad\quad$ and $\forall i \in I. \; [\, s \mid v{:}\, i \,], Hi \vDash p.$

—

Axiom Schemata

$$m > n \Rightarrow \left(\bigodot_{i=m}^{n} p(i) \Leftrightarrow \mathbf{emp}\right)$$

$$m = n \Rightarrow \left(\bigodot_{i=m}^{n} p(i) \Leftrightarrow p(m)\right)$$

$$k \leq m \leq n + 1 \Rightarrow \left(\bigodot_{i=k}^{n} p(i) \Leftrightarrow \left(\bigodot_{i=k}^{m-1} p(i) \ * \ \bigodot_{i=m}^{n} p(i)\right)\right)$$

$$\bigodot_{i=m}^{n} p(i) \Leftrightarrow \bigodot_{i=m-k}^{n-k} p(i + k)$$

$$m \leq n \Rightarrow \left(\left(\bigodot_{i=m}^{n} p(i)\right) \ * \ q \Leftrightarrow \bigodot_{i=m}^{n} (p(i) \ * \ q)\right)$$

when q is pure and $i \notin \mathsf{FV}(q)$

$$m \leq n \Rightarrow \left(\left(\bigodot_{i=m}^{n} p(i)\right) \wedge q \Leftrightarrow \bigodot_{i=m}^{n} (p(i) \wedge q)\right)$$

when q is pure and $i \notin \mathsf{FV}(q)$

$$m \leq j \leq n \Rightarrow \left(\left(\bigodot_{i=m}^{n} p(i)\right) \Rightarrow (p(j) \ * \ \mathbf{true})\right)$$

—

Array Allocation

$$\langle \text{comm} \rangle ::= \cdots \mid \langle \text{var} \rangle := \textbf{allocate}\ \langle \text{exp} \rangle$$

$$\text{x} := \textbf{allocate}\ \text{y} \qquad \begin{array}{ll} \text{Store}: & \text{x}: 3,\ \text{y}: 4 \\ \text{Heap}: & \text{empty} \\ & \Downarrow \\ \text{Store}: & \text{x}: 37,\ \text{y}: 4 \\ \text{Heap}: & 37: -,\ 38: -,\ 39: -,\ 40: - \end{array}$$

Nonoverwriting Inference Rules

- The local nonoverwriting form (ALLOCNOL)

  $$\frac{}{\{\textbf{emp}\}\ v := \textbf{allocate}\ e\ \{\bigodot_{i=v}^{v+e-1} i \mapsto -\},}$$

  where $v \notin \text{FV}(e)$.

- The global nonoverwriting form (ALLOCNOG)

  $$\frac{}{\{r\}\ v := \textbf{allocate}\ e\ \{(\bigodot_{i=v}^{v+e-1} i \mapsto -)\ *\ r\},}$$

  where $v \notin \text{FV}(e, r)$.

—

General Inference Rules

- The local form (ALLOCL)

$$\frac{}{\{v = v' \wedge \mathbf{emp}\}\ v := \mathbf{allocate}\ e\ \{\bigodot_{i=v}^{v+e'-1} i \mapsto -\},}$$

where v' is distinct from v, and e' denotes $e/v \rightarrow v'$.

- The global form (ALLOCG)

$$\frac{}{\{r\}\ v := \mathbf{allocate}\ e\ \{\exists v'.\ (\bigodot_{i=v}^{v+e'-1} i \mapsto -) \ast r'\},}$$

where v' is distinct from v, $v' \notin \mathsf{FV}(e, r)$, e' denotes $e/v \rightarrow v'$, and r' denotes $r/v \rightarrow v'$.

- The backward-reasoning form (ALLOCBR)

$$\frac{}{\{\forall v''.\ (\bigodot_{i=v''}^{v''+e-1} i \mapsto -) -\!\!\ast\ p''\}\ v := \mathbf{allocate}\ e\ \{p\},}$$

where v'' is distinct from v, $v'' \notin \mathsf{FV}(e, p)$, and p'' denotes $p/v \rightarrow v''$.

—

Arrays that Denote Sequences

$$\text{array}\ \alpha\ (a, b) \stackrel{\text{def}}{=} \#\alpha = b - a + 1 \wedge \bigodot_{i=a}^{b} i \mapsto \alpha_{i-a+1}.$$

(Since the length of a sequence is never negative, the assertion $\text{array}\ \alpha\ (a, b)$ implies that $a \leq b + 1$.)

Properties

$$\text{array}\ \alpha\ (a, b) \Rightarrow \#\alpha = b - a + 1$$

$$\text{array}\ \alpha\ (a, b) \Rightarrow i \hookrightarrow \alpha_{i-a+1} \quad \text{when } a \leq i \leq b$$

$$\text{array}\ \epsilon\ (a, b) \Leftrightarrow b = a - 1 \wedge \mathbf{emp}$$

$$\text{array}\ x\ (a, b) \Leftrightarrow b = a \wedge a \mapsto x$$

$$\text{array}\ x{\cdot}\alpha\ (a, b) \Leftrightarrow a \mapsto x \ * \ \text{array}\ \alpha\ (a + 1, b)$$

$$\text{array}\ \alpha{\cdot}x\ (a, b) \Leftrightarrow \text{array}\ \alpha\ (a, b - 1) \ * \ b \mapsto x$$

$$\begin{aligned} \text{array}\ \alpha\ (a, c) \ * \ \text{array}\ \beta\ (c + 1, b) & \\ \Leftrightarrow \text{array}\ \alpha{\cdot}\beta\ (a, b) \wedge c = a + \#\alpha - 1 & \\ \Leftrightarrow \text{array}\ \alpha{\cdot}\beta\ (a, b) \wedge c = b - \#\beta & \end{aligned}$$

—

Partition

$\{\mathsf{array}\ \alpha(\mathsf{a},\mathsf{b})\}$

$\textbf{newvar}\ \mathsf{d},\mathsf{x},\mathsf{y}\ \textbf{in}\ \big(\mathsf{c} := \mathsf{a}-1\ ;\ \mathsf{d} := \mathsf{b}+1\ ;$

$\{\exists\alpha_1,\alpha_2,\alpha_3.\ (\mathsf{array}\ \alpha_1\,(\mathsf{a},\mathsf{c})\ *\ \mathsf{array}\ \alpha_2\,(\mathsf{c}+1,\mathsf{d}-1)$
$\quad *\ \mathsf{array}\ \alpha_3\,(\mathsf{d},\mathsf{b}))\wedge \alpha_1{\cdot}\alpha_2{\cdot}\alpha_3 \sim \alpha \wedge \{\alpha_1\} \leq^* \mathsf{r} \wedge \{\alpha_3\} >^* \mathsf{r}\}$

$\textbf{while}\ \mathsf{d} > \mathsf{c}+1\ \textbf{do}\ \big(\mathsf{x} := [\mathsf{c}+1];$

$\textbf{if}\ \mathsf{x} \leq \mathsf{r}\ \textbf{then}$

$\{\exists\alpha_1,\alpha_2,\alpha_3.\ (\mathsf{array}\ \alpha_1\,(\mathsf{a},\mathsf{c})\ *\ \mathsf{c}{+}1 \mapsto \mathsf{x} * \mathsf{array}\ \alpha_2\,(\mathsf{c}{+}2,\mathsf{d}{-}1)$
$\ *\ \mathsf{array}\ \alpha_3\,(\mathsf{d},\mathsf{b}))\wedge \alpha_1{\cdot}\mathsf{x}{\cdot}\alpha_2{\cdot}\alpha_3 \sim \alpha \wedge \{\alpha_1{\cdot}\mathsf{x}\} \leq^* \mathsf{r} \wedge \{\alpha_3\} >^* \mathsf{r}\}$

$\mathsf{c} := \mathsf{c}+1$

$\textbf{else}\ \big(\mathsf{y} := [\mathsf{d}-1];$

$\textbf{if}\ \mathsf{y} > \mathsf{r}\ \textbf{then}$

$\{\exists\alpha_1,\alpha_2,\alpha_3.\ (\mathsf{array}\ \alpha_1\,(\mathsf{a},\mathsf{c}) * \mathsf{array}\ \alpha_2\,(\mathsf{c}{+}1,\mathsf{d}{-}2) * \mathsf{d}{-}1 \mapsto \mathsf{y}$
$\ *\ \mathsf{array}\ \alpha_3\,(\mathsf{d},\mathsf{b}))\wedge \alpha_1{\cdot}\alpha_2{\cdot}\mathsf{y}{\cdot}\alpha_3 \sim \alpha \wedge \{\alpha_1\} \leq^* \mathsf{r} \wedge \{\mathsf{y}{\cdot}\alpha_3\} >^* \mathsf{r}\}$

$\mathsf{d} := \mathsf{d}-1$

$\textbf{else}$

$\{\exists\alpha_1,\alpha_2,\alpha_3.\ (\mathsf{array}\ \alpha_1\,(\mathsf{a},\mathsf{c})\ *\ \mathsf{c}+1 \mapsto \mathsf{x} \qquad (*)$
$\quad *\ \mathsf{array}\ \alpha_2\,(\mathsf{c}+2,\mathsf{d}-2)\ *\ \mathsf{d}-1 \mapsto \mathsf{y}\ *\ \mathsf{array}\ \alpha_3\,(\mathsf{d},\mathsf{b}))$
$\quad \wedge\ \alpha_1{\cdot}\mathsf{x}{\cdot}\alpha_2{\cdot}\mathsf{y}{\cdot}\alpha_3 \sim \alpha \wedge \{\alpha_1\} \leq^* \mathsf{r} \wedge \{\alpha_3\} >^* \mathsf{r} \wedge \mathsf{x} > \mathsf{r} \wedge \mathsf{y} \leq \mathsf{r}\}$

$([\mathsf{c}+1] := \mathsf{y}\ ;\ [\mathsf{d}-1] := \mathsf{x}\ ;\ \mathsf{c} := \mathsf{c}+1\ ;\ \mathsf{d} := \mathsf{d}-1)\big)\big)\big)$

$\{\exists\alpha_1,\alpha_2.\ (\mathsf{array}\ \alpha_1(\mathsf{a},\mathsf{c})\ *\ \mathsf{array}\ \alpha_2(\mathsf{c}+1,\mathsf{b}))$
$\ \wedge\ \alpha_1{\cdot}\alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \mathsf{r} \wedge \mathsf{r} <^* \{\alpha_2\}\}$

—

A Subtlety

In the assertion marked $(*)$:

$$\{\exists \alpha_1, \alpha_2, \alpha_3.\ (\mathsf{array}\ \alpha_1\ (\mathsf{a}, \mathsf{c})\ *\ \mathsf{c}+1 \mapsto \mathsf{x}$$
$$*\ \mathsf{array}\ \alpha_2\ (\mathsf{c}+2, \mathsf{d}-2)\ *\ \mathsf{d}-1 \mapsto \mathsf{y}\ *\ \mathsf{array}\ \alpha_3\ (\mathsf{d}, \mathsf{b}))$$
$$\wedge\ \alpha_1{\cdot}\mathsf{x}{\cdot}\alpha_2{\cdot}\mathsf{y}{\cdot}\alpha_3 \sim \alpha \wedge \{\alpha_1\} \leq^* \mathsf{r} \wedge \{\alpha_3\} >^* \mathsf{r} \wedge \mathsf{x} > \mathsf{r} \wedge \mathsf{y} \leq \mathsf{r}\}$$

it is the **while**-test $\mathsf{d} > \mathsf{c} + 1$, plus

$$\mathsf{c}+1 \hookrightarrow \mathsf{x} \wedge \mathsf{d}-1 \hookrightarrow \mathsf{y} \wedge \mathsf{x} > \mathsf{r} \wedge \mathsf{y} \leq \mathsf{r} \Rightarrow \mathsf{c}+1 \neq \mathsf{d}-1,$$

that guarantees that $\mathsf{c}+1 < \mathsf{d}-1$, so that

$$\mathsf{array}\ \alpha_2\ (\mathsf{c}+2, \mathsf{d}-2)$$

makes sense.

—

From Partition to Quicksort

If we define

$$
\begin{array}{l}
\mathsf{partition}(\mathsf{c};\mathsf{a},\mathsf{b},\mathsf{r}) = \\
\quad \mathbf{newvar}\ \mathsf{d},\mathsf{x},\mathsf{y}\ \mathbf{in}\ \big(\mathsf{c} := \mathsf{a} - 1\ ;\ \mathsf{d} := \mathsf{b} + 1\ ; \\
\quad \mathbf{while}\ \mathsf{d} > \mathsf{c} + 1\ \mathbf{do} \\
\qquad \big(\mathsf{x} := [\mathsf{c} + 1]\ ;\ \mathbf{if}\ \mathsf{x} \leq \mathsf{r}\ \mathbf{then}\ \mathsf{c} := \mathsf{c} + 1\ \mathbf{else} \\
\qquad\ \big(\mathsf{y} := [\mathsf{d} - 1]\ ;\ \mathbf{if}\ \mathsf{y} > \mathsf{r}\ \mathbf{then}\ \mathsf{d} := \mathsf{d} - 1\ \mathbf{else} \\
\qquad\quad ([\mathsf{c} + 1] := \mathsf{y}\ ;\ [\mathsf{d} - 1] := \mathsf{x}\ ;\ \mathsf{c} := \mathsf{c} + 1\ ;\ \mathsf{d} := \mathsf{d} - 1)\big)\big)\big),
\end{array}
$$

then

$$
\begin{array}{l}
\{\mathsf{array}\ \alpha(\mathsf{a},\mathsf{b})\} \\
\mathsf{partition}(\mathsf{c};\mathsf{a},\mathsf{b},\mathsf{r})\{\alpha\} \\
\{\exists \alpha_1, \alpha_2.\ (\mathsf{array}\ \alpha_1(\mathsf{a},\mathsf{c})\ *\ \mathsf{array}\ \alpha_2(\mathsf{c}+1,\mathsf{b})) \\
\quad \wedge\ \alpha_1{\cdot}\alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \mathsf{r} \wedge \mathsf{r} <^* \{\alpha_2\}\}.
\end{array}
$$

Then we can use partition to define a procedure satisfying

$$
\begin{array}{l}
\{\mathsf{array}\ \alpha\ (\mathsf{a},\mathsf{b})\} \\
\mathsf{quicksort}(;\mathsf{a},\mathsf{b})\{\alpha\} \\
\{\exists \beta.\ \mathsf{array}\ \beta\ (\mathsf{a},\mathsf{b}) \wedge \beta \sim \alpha \wedge \mathbf{ord}\ \beta\}.
\end{array}
$$

—

Quicksort (continued)

$$\{\mathsf{array}\ \alpha\ (\mathsf{a},\mathsf{b})\}$$

$$\mathbf{if}\ \mathsf{a} < \mathsf{b}\ \mathbf{then}\ \mathbf{newvar}\ \mathsf{c}\ \mathbf{in}$$

$$\big(\{\exists \mathsf{x}_1, \alpha_0, \mathsf{x}_2.\ (\mathsf{a} \mapsto \mathsf{x}_1\ *\ \mathsf{array}\ \alpha_0(\mathsf{a}+1, \mathsf{b}-1)\ *\ \mathsf{b} \mapsto \mathsf{x}_2)$$
$$\wedge\ \mathsf{x}_1{\cdot}\alpha_0{\cdot}\mathsf{x}_2 \sim \alpha\}$$

$$\mathbf{newvar}\ \mathsf{x1}, \mathsf{x2}, \mathsf{r}\ \mathbf{in}$$

$$\big(\mathsf{x1} := [\mathsf{a}]\ ;\ \mathsf{x2} := [\mathsf{b}]\ ;$$

$$\mathbf{if}\ \mathsf{x1} > \mathsf{x2}\ \mathbf{then}\ ([\mathsf{a}] := \mathsf{x2}\ ;\ [\mathsf{b}] := \mathsf{x1})\ \mathbf{else\ skip}\ ;$$

$$\mathsf{r} := (\mathsf{x1} + \mathsf{x2}) \div 2\ ;$$

$$\{\exists \mathsf{x}_1, \alpha_0, \mathsf{x}_2.\ (\mathsf{a} \mapsto \mathsf{x}_1\ *\ \mathsf{array}\ \alpha_0(\mathsf{a}+1, \mathsf{b}-1)\ *\ \mathsf{b} \mapsto \mathsf{x}_2)$$
$$\wedge\ \mathsf{x}_1{\cdot}\alpha_0{\cdot}\mathsf{x}_2 \sim \alpha \wedge \mathsf{x}_1 \le \mathsf{r} \le \mathsf{x}_2\}$$

$$\left.\begin{array}{l} \{\mathsf{array}\ \alpha_0(\mathsf{a}+1, \mathsf{b}-1)\} \\ \mathsf{partition}(\mathsf{c}; \mathsf{a}{+}1, \mathsf{b}{-}1, \mathsf{r})\{\alpha_0\} \\ \{\exists \alpha_1, \alpha_2.\ (\mathsf{array}\ \alpha_1(\mathsf{a}+1, \mathsf{c}) \\ \quad *\ \mathsf{array}\ \alpha_2(\mathsf{c}+1, \mathsf{b}-1)) \\ \ \wedge\ \alpha_1{\cdot}\alpha_2 \sim \alpha_0 \\ \ \wedge\ \{\alpha_1\} \le^* \mathsf{r} \wedge \mathsf{r} <^* \{\alpha_2\}\} \end{array}\right\} * \left(\begin{array}{l} (\mathsf{a}{\mapsto}\mathsf{x}_1 * \mathsf{b}{\mapsto}\mathsf{x}_2) \\ \ \wedge\ \mathsf{x}_1{\cdot}\alpha_0{\cdot}\mathsf{x}_2 \sim \alpha \\ \ \wedge\ \mathsf{x}_1 \le \mathsf{r} \le \mathsf{x}_2 \end{array}\right)\Bigg\}\ \exists \mathsf{x}_1, \alpha_0, \mathsf{x}_2$$

$$\{\exists \mathsf{x}_1, \alpha_1, \alpha_2, \mathsf{x}_2.$$
$$(\mathsf{a} \mapsto \mathsf{x}_1 * \mathsf{array}\ \alpha_1(\mathsf{a}+1, \mathsf{c}) * \mathsf{array}\ \alpha_2(\mathsf{c}+1, \mathsf{b}-1) * \mathsf{b} \mapsto \mathsf{x}_2)$$
$$\wedge\ \mathsf{x}_1{\cdot}\alpha_1{\cdot}\alpha_2{\cdot}\mathsf{x}_2 \sim \alpha \wedge \mathsf{x}_1 \le \mathsf{r} \le \mathsf{x}_2 \wedge \{\alpha_1\} \le^* \mathsf{r} \wedge \mathsf{r} <^* \{\alpha_2\}\}\big)\ ;$$

$$\{\exists \alpha_1, \alpha_2.\ (\mathsf{array}\ \alpha_1\ (\mathsf{a}, \mathsf{c})\ *\ \mathsf{array}\ \alpha_2\ (\mathsf{c}+1, \mathsf{b}))$$
$$\wedge\ \alpha_1{\cdot}\alpha_2 \sim \alpha \wedge \{\alpha_1\} \le^* \{\alpha_2\}\}$$

$\vdots$

—

$\vdots$

$$\{\exists\alpha_1,\alpha_2.\ (\mathsf{array}\ \alpha_1\ (\mathsf{a},\mathsf{c})\ *\ \mathsf{array}\ \alpha_2\ (\mathsf{c}+1,\mathsf{b}))$$
$$\wedge\ \alpha_1{\cdot}\alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\}\}$$

$$\left.\begin{array}{l}\{\mathsf{array}\ \alpha_1\ (\mathsf{a},\mathsf{c})\}\\ \mathsf{quicksort}(;\mathsf{a},\mathsf{c})\{\alpha_1\}\\ \{\exists\beta.\ \mathsf{array}\ \beta\ (\mathsf{a},\mathsf{c})\\ \quad\wedge\ \beta \sim \alpha_1 \wedge \mathbf{ord}\ \beta\}\end{array}\right\} * \left(\begin{array}{l}\mathsf{array}\ \alpha_2(\mathsf{c}+1,\mathsf{b})\\ \quad\wedge\ \alpha_1{\cdot}\alpha_2 \sim \alpha\\ \quad\wedge\ \{\alpha_1\} \leq^* \{\alpha_2\}\end{array}\right)\Bigg\}\ \exists\alpha_1,\exists\alpha_2$$

$$\{\exists\beta_1,\alpha_2.\ (\mathsf{array}\ \beta_1\ (\mathsf{a},\mathsf{c})\ *\ \mathsf{array}\ \alpha_2\ (\mathsf{c}+1,\mathsf{b}))$$
$$\wedge\ \beta_1{\cdot}\alpha_2 \sim \alpha \wedge \{\beta_1\} \leq^* \{\alpha_2\} \wedge \mathbf{ord}\ \beta_1\}$$

$$\left.\begin{array}{l}\{\mathsf{array}\ \alpha_2\ (\mathsf{c}+1,\mathsf{b})\}\\ \mathsf{quicksort}(;\mathsf{c}+1,\mathsf{b})\{\alpha_2\}\\ \{\exists\beta.\ \mathsf{array}\ \beta\ (\mathsf{c}+1,\mathsf{b})\\ \quad\wedge\ \beta \sim \alpha_2 \wedge \mathbf{ord}\ \beta\}\end{array}\right\} * \left(\begin{array}{l}\mathsf{array}\ \beta_1(\mathsf{a},\mathsf{c})\\ \quad\wedge\ \beta_1{\cdot}\alpha_2 \sim \alpha\\ \quad\wedge\ \{\beta_1\} \leq^* \{\alpha_2\}\\ \quad\wedge\ \mathbf{ord}\ \beta_1\end{array}\right)\Bigg\}\ \exists\beta_1,\exists\alpha_2$$

$$\{\exists\beta_1,\beta_2.\ (\mathsf{array}\ \beta_1\ (\mathsf{a},\mathsf{c})\ *\ \mathsf{array}\ \beta_2\ (\mathsf{c}+1,\mathsf{b}))$$
$$\wedge\ \beta_1{\cdot}\beta_2 \sim \alpha \wedge \{\beta_1\} \leq^* \{\beta_2\} \wedge \mathbf{ord}\ \beta_1 \wedge \mathbf{ord}\ \beta_2\}\Big)$$

else skip

$$\{\exists\beta.\ \mathsf{array}\ \beta\ (\mathsf{a},\mathsf{b}) \wedge \beta \sim \alpha \wedge \mathbf{ord}\ \beta\}$$

Thus we may define:

$$\begin{array}{l}\mathsf{quicksort}(\mathsf{a},\mathsf{b}) =\\ \quad \mathbf{if}\ \mathsf{a} < \mathsf{b}\ \mathbf{then\ newvar}\ \mathsf{c}\ \mathbf{in}\\ \qquad \Big(\mathbf{newvar}\ \mathsf{x1},\mathsf{x2},\mathsf{r}\ \mathbf{in}\\ \qquad\quad \Big(\mathsf{x1} := [\mathsf{a}]\ ;\ \mathsf{x2} := [\mathsf{b}]\ ;\\ \qquad\quad \mathbf{if}\ \mathsf{x1} > \mathsf{x2}\ \mathbf{then}\ ([\mathsf{a}] := \mathsf{x2}\ ;\ [\mathsf{b}] := \mathsf{x1})\ \mathbf{else\ skip}\ ;\\ \qquad\quad \mathsf{r} := (\mathsf{x1}+\mathsf{x2}) \div 2\ ;\ \mathsf{partition}(\mathsf{a}+1,\mathsf{b}-1,\mathsf{r};\mathsf{c})\Big)\ ;\\ \qquad \mathsf{quicksort}(\mathsf{a},\mathsf{c})\ ;\ \mathsf{quicksort}(\mathsf{c}+1,\mathsf{b})\Big)\\ \quad \mathbf{else\ skip}.\end{array}$$

—

A Cyclic Buffer Using an Array

We assume that an n-element array has been allocated at location l, and we write $x \oplus y$ for the integer such that

$$x \oplus y = x + y \text{ modulo } \mathsf{n} \quad \text{and} \quad \mathsf{l} \leq \mathsf{j} < \mathsf{l} + \mathsf{n}.$$

We will use the variables

m	number of active elements
i	pointer to first active element
j	pointer to first inactive element

Let R abbreviate the assertion

$$R \stackrel{\text{def}}{=} 0 \leq \mathsf{m} \leq \mathsf{n} \wedge \mathsf{l} \leq \mathsf{i} < \mathsf{l} + \mathsf{n} \wedge \mathsf{l} \leq \mathsf{j} < \mathsf{l} + \mathsf{n} \wedge \mathsf{j} = \mathsf{i} \oplus \mathsf{m}$$

It is easy to show that

$$\{R \wedge \mathsf{m} < \mathsf{n}\}$$
$$\mathsf{m} := \mathsf{m} + 1 \,;\, \mathbf{if}\ \mathsf{j} = \mathsf{l} + \mathsf{n} - 1\ \mathbf{then}\ \mathsf{j} := \mathsf{l}\ \mathbf{else}\ \mathsf{j} := \mathsf{j} + 1$$
$$\{R\}$$

and

$$\{R \wedge \mathsf{m} > 0\}$$
$$\mathsf{m} := \mathsf{m} - 1 \,;\, \mathbf{if}\ \mathsf{i} = \mathsf{l} + \mathsf{n} - 1\ \mathbf{then}\ \mathsf{i} := \mathsf{l}\ \mathbf{else}\ \mathsf{i} := \mathsf{i} + 1$$
$$\{R\}$$

When the buffer contains a sequence α, it should satisfy

$$((\bigodot_{\mathsf{k}=0}^{\mathsf{m}-1} \mathsf{i} \oplus \mathsf{k} \mapsto \alpha_{\mathsf{k}+1}) * (\bigodot_{\mathsf{k}=0}^{\mathsf{n}-\mathsf{m}-1} \mathsf{j} \oplus \mathsf{k} \mapsto -)) \wedge \mathsf{m} = \#\alpha \wedge R.$$

—

Inserting an Element

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto \alpha_{k+1}) \, * \, (\bigodot_{k=0}^{n-m-1} j \oplus k \mapsto -)) \\ \wedge m = \#\alpha \wedge R \wedge m < n\}$$

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto \alpha_{k+1}) \, * \, (\bigodot_{k=0}^{0} j \oplus k \mapsto -) \, * \\ (\bigodot_{k=1}^{n-m-1} j \oplus k \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto \alpha_{k+1}) \, * \, j \oplus 0 \mapsto - \, * \\ (\bigodot_{k=1}^{n-m-1} j \oplus k \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$

$[j] := x \, ;$

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto \alpha_{k+1}) \, * \, j \oplus 0 \mapsto x \, * \\ (\bigodot_{k=1}^{n-m-1} j \oplus k \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto \alpha_{k+1}) \, * \, i \oplus m \mapsto x \, * \\ (\bigodot_{k=1}^{n-m-1} j \oplus k \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto (\alpha{\cdot}x)_{k+1}) \, * \, i \oplus m \mapsto (\alpha{\cdot}x)_{m+1} \, * \\ (\bigodot_{k=1}^{n-m-1} j \oplus k \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto (\alpha{\cdot}x)_{k+1}) \, * \, (\bigodot_{k=m}^{m} i \oplus k \mapsto (\alpha{\cdot}x)_{k+1}) \, * \\ (\bigodot_{k=1}^{n-m-1} j \oplus k \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$

$$\{(((\bigodot_{k=0}^{m} i \oplus k \mapsto (\alpha{\cdot}x)_{k+1}) \, * \, (\bigodot_{k=1}^{n-m-1} j \oplus k \mapsto -)) \\ \wedge m + 1 = \#(\alpha{\cdot}x) \wedge R \wedge m < n\}$$

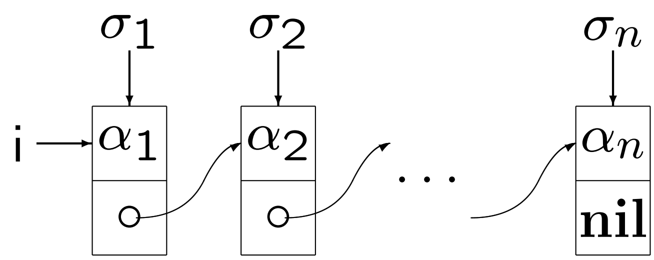
$$\{(((\bigodot_{k=0}^{m} i \oplus k \mapsto (\alpha{\cdot}x)_{k+1}) \, * \, (\bigodot_{k=0}^{n-m-2} j \oplus k \oplus 1 \mapsto -)) \\ \wedge m + 1 = \#(\alpha{\cdot}x) \wedge R \wedge m < n\}$$

$m := m + 1 \, ; \, \mathbf{if} \; j = l + n - 1 \; \mathbf{then} \; j := l \; \mathbf{else} \; j := j + 1$

$$\{(((\bigodot_{k=0}^{m-1} i \oplus k \mapsto (\alpha{\cdot}x)_{k+1}) \, * \, (\bigodot_{k=0}^{n-m-1} j \oplus k \mapsto -)) \\ \wedge m = \#(\alpha{\cdot}x) \wedge R\}$$

—

Connecting Two Views of Lists



If

$$\mathsf{list}\ \epsilon\ \mathsf{i} \stackrel{\mathrm{def}}{=} \mathbf{emp} \wedge \mathsf{i} = \mathbf{nil}$$

$$\mathsf{list}\ (\mathsf{a}{\cdot}\alpha)\ \mathsf{i} \stackrel{\mathrm{def}}{=} \exists \mathsf{j}.\ \mathsf{i} \mapsto \mathsf{a}, \mathsf{j}\ *\ \mathsf{list}\ \alpha\ \mathsf{j}$$

and

$$\mathsf{listN}\ \epsilon\ \mathsf{i} \stackrel{\mathrm{def}}{=} \mathbf{emp} \wedge \mathsf{i} = \mathbf{nil}$$

$$\mathsf{listN}\ (\mathsf{b}{\cdot}\sigma)\ \mathsf{i} \stackrel{\mathrm{def}}{=} \mathsf{b} = \mathsf{i} \wedge \exists \mathsf{j}.\ \mathsf{i} + 1 \mapsto \mathsf{j}\ *\ \mathsf{listN}\ \sigma\ \mathsf{j},$$

then

$$\mathsf{list}\ \alpha\ \mathsf{i} \Leftrightarrow \exists \sigma.\ \#\sigma = \#\alpha \wedge (\mathsf{listN}\ \sigma\ \mathsf{i}\ *\ \bigodot_{\mathsf{k}=1}^{\#\alpha} \sigma_k \mapsto \alpha_k).$$

The proof is by induction on α.

—

Specifying Subset Lists

We use the following variables to denote various kinds of sequences:

$$\begin{aligned} \alpha &: \text{ sequences of integers} \\ \beta, \gamma &: \text{ nonempty sequences of addresses} \\ \sigma &: \text{ nonempty sequences of sequences of integers.} \end{aligned}$$

Our goal is to write a procedure $\mathsf{subsets}$ satisfying

$$\begin{aligned} &H_{\mathsf{subsets}} \overset{\mathsf{def}}{=} \\ &\quad \{\mathsf{list}\,\alpha\,\mathsf{i}\} \\ &\quad \mathsf{subsets}(\mathsf{j};\mathsf{i})\{\alpha\} \\ &\quad \{\exists\sigma,\beta.\ \ \mathsf{ss}(\alpha,\sigma) \land \big(\mathsf{list}\,\alpha\,\mathsf{i} \ * \ \mathsf{list}\,\beta\,\mathsf{j} \ * \ (Q(\sigma,\beta) \land R(\beta))\big)\}, \end{aligned}$$

where

$$\begin{aligned} \#\mathsf{ext_a}\sigma &\overset{\mathsf{def}}{=} \#\sigma \\ (\mathsf{ext_a}\sigma)_i &\overset{\mathsf{def}}{=} \mathsf{a}{\cdot}\sigma_i \\ \mathsf{ss}(\epsilon,\sigma) &\overset{\mathsf{def}}{=} \sigma = [\epsilon] \\ \mathsf{ss}(\mathsf{a}{\cdot}\alpha,\sigma) &\overset{\mathsf{def}}{=} \exists\sigma'.\ \ \mathsf{ss}(\alpha,\sigma') \land \sigma = (\mathsf{ext_a}\sigma'){\cdot}\sigma' \\ Q(\sigma,\beta) &\overset{\mathsf{def}}{=} \#\beta = \#\sigma \land \forall_{\mathsf{i}=1}^{\#\beta}(\mathsf{list}\,\sigma_{\mathsf{i}}\,\beta_{\mathsf{i}} \ * \ \mathbf{true}) \\ R(\beta) &\overset{\mathsf{def}}{=} (\beta_{\#\beta} = \mathbf{nil} \land \mathbf{emp}) \ * \\ &\qquad\qquad \bigodot_{\mathsf{i}=1}^{\#\beta-1}(\exists\mathsf{a},\mathsf{k}.\ \ \mathsf{i} < \mathsf{k} \le \#\beta \land \beta_{\mathsf{i}} \mapsto \mathsf{a},\beta_{\mathsf{k}}). \end{aligned}$$

—

The Storage Used by subsets

By induction on the definition of ss,

$$\mathsf{ss}(\epsilon, \sigma) \stackrel{\text{def}}{=} \sigma = [\epsilon]$$
$$\mathsf{ss}(\mathsf{a}{\cdot}\alpha, \sigma) \stackrel{\text{def}}{=} \exists \sigma'.\ \ \mathsf{ss}(\alpha, \sigma') \wedge \sigma = (\mathsf{ext}_{\mathsf{a}}\sigma'){\cdot}\sigma',$$

using $\#\mathsf{ext}_{\mathsf{a}}\sigma = \#\sigma$:

$$\mathsf{ss}(\alpha, \sigma) \Rightarrow \#\sigma = 2^{\#\alpha}.$$

By the definition of Q,

$$Q(\sigma, \beta) \stackrel{\text{def}}{=} \#\beta = \#\sigma \wedge \forall_{\mathsf{i}=1}^{\#\beta}(\mathsf{list}\ \sigma_{\mathsf{i}}\ \beta_{\mathsf{i}}\ *\ \mathbf{true}),$$

we have

$$\#\beta = \#\sigma.$$

By induction on the definition of list:

list α describes a heap containing $\#\alpha$ two-cells,

list β describes a heap containing $\#\beta$ two-cells.

By the definition of $R(\beta)$:

$$R(\beta) \stackrel{\text{def}}{=} (\beta_{\#\beta} = \mathbf{nil} \wedge \mathbf{emp})\ *$$
$$\bigodot_{\mathsf{i}=1}^{\#\beta-1}(\exists \mathsf{a}, \mathsf{k}.\ \ \mathsf{i} < \mathsf{k} \leq \#\beta \wedge \beta_{\mathsf{i}} \mapsto \mathsf{a}, \beta_{\mathsf{k}}),$$

and of $\bigodot$:

$R(\beta)$ describes a heap containing $\#\beta - 1$ two-cells.

—

The Storage Used by subsets (continued)

Thus the postcondition of the specification of subsets:

$$\{\exists \sigma, \beta.\ \mathsf{ss}(\alpha, \sigma) \wedge \big(\underline{\mathsf{list}\ \alpha\ \mathsf{i}} * \underline{\mathsf{list}\ \beta\ \mathsf{j}} * (Q(\sigma, \beta) \wedge \underline{R(\beta)})\big)\}$$

describes a heap containing three disjoint parts:

- a list containing $\#\alpha$ two-cells (the input list),
- a list containing $2^{\#\alpha}$ two-cells,
- a list containing $2^{\#\alpha} - 1$ two-cells.

—

Some Properties

The predicates

$$Q(\sigma, \beta) \stackrel{\text{def}}{=} \#\beta = \#\sigma \wedge \forall_{\mathsf{i}=1}^{\#\beta}(\mathsf{list}\ \sigma_{\mathsf{i}}\ \beta_{\mathsf{i}}\ *\ \mathbf{true})$$

$$R(\beta) \stackrel{\text{def}}{=} (\beta_{\#\beta} = \mathbf{nil} \wedge \mathbf{emp})\ *$$
$$\bigodot_{\mathsf{i}=1}^{\#\beta-1}(\exists \mathsf{a}, \mathsf{k}.\ \ \mathsf{i} < \mathsf{k} \leq \#\beta \wedge \beta_{\mathsf{i}} \mapsto \mathsf{a}, \beta_{\mathsf{k}})$$

$$W(\beta, \gamma, \mathsf{a}) \stackrel{\text{def}}{=} \#\gamma = \#\beta \wedge \bigodot_{\mathsf{i}=1}^{\#\gamma} \gamma_{\mathsf{i}} \mapsto \mathsf{a}, \beta_{\mathsf{i}}$$

satisfy

$$Q([\epsilon], [\mathbf{nil}]) \Leftrightarrow \mathbf{true}$$

$$R([\mathbf{nil}]) \Leftrightarrow \mathbf{emp}$$

$$W(\beta, \gamma, \mathsf{a})\ *\ \mathsf{g} \mapsto \mathsf{a}, \mathsf{b} \Leftrightarrow W(\mathsf{b}\cdot\beta, \mathsf{g}\cdot\gamma, \mathsf{a}) \qquad (1)$$

$$Q(\sigma, \beta)\ *\ W(\beta, \gamma, \mathsf{a}) \Rightarrow Q((\mathsf{ext}_{\mathsf{a}}\sigma)\cdot\sigma, \gamma\cdot\beta) \qquad (2)$$

$$R(\beta)\ *\ W(\beta, \gamma, \mathsf{a}) \Rightarrow R(\gamma\cdot\beta) \qquad (3)$$

$$(Q(\sigma, \beta) \wedge R(\beta))\ *\ W(\beta, \gamma, \mathsf{a}) \Rightarrow Q((\mathsf{ext}_{\mathsf{a}}\sigma)\cdot\sigma, \gamma\cdot\beta) \wedge R(\gamma\cdot\beta).$$

—

Proofs (1)

$$
\begin{aligned}
& W(\beta, \gamma, \mathsf{a}) \ * \ \mathsf{g} \mapsto \mathsf{a}, \mathsf{b} \\
& \quad \Leftrightarrow \#\gamma = \#\beta \wedge \left(\mathsf{g} \mapsto \mathsf{a}, \mathsf{b} \ * \ \bigodot_{\mathsf{i}=1}^{\#\gamma} \gamma_{\mathsf{i}} \mapsto \mathsf{a}, \beta_{\mathsf{i}}\right) \\
& \quad \Leftrightarrow \#\mathsf{g}{\cdot}\gamma = \#\mathsf{b}{\cdot}\beta \wedge \Big(\Big(\bigodot_{\mathsf{i}=1}^{1} (\mathsf{g}{\cdot}\gamma)_{\mathsf{i}} \mapsto \mathsf{a}, (\mathsf{b}{\cdot}\beta)_{\mathsf{i}}\Big) \ * \\
& \qquad \Big(\bigodot_{\mathsf{i}=1}^{\#\mathsf{g}\cdot\gamma-1} (\mathsf{g}{\cdot}\gamma)_{\mathsf{i}+1} \mapsto \mathsf{a}, (\mathsf{b}{\cdot}\beta)_{\mathsf{i}+1}\Big)\Big) \\
& \quad \Leftrightarrow \#\mathsf{g}{\cdot}\gamma = \#\mathsf{b}{\cdot}\beta \wedge \bigodot_{\mathsf{i}=1}^{\#\mathsf{g}\cdot\gamma} (\mathsf{g}{\cdot}\gamma)_{\mathsf{i}} \mapsto \mathsf{a}, (\mathsf{b}{\cdot}\beta)_{\mathsf{i}} \\
& \quad \Leftrightarrow W(\mathsf{b}{\cdot}\beta, \mathsf{g}{\cdot}\gamma, \mathsf{a}).
\end{aligned}
$$

—

Proofs (2)

Let

$$p(\mathsf{i}) \stackrel{\mathsf{def}}{=} \mathsf{list}\,\sigma_\mathsf{i}\,\beta_\mathsf{i} \qquad q(\mathsf{i}) \stackrel{\mathsf{def}}{=} \gamma_\mathsf{i} \mapsto \mathsf{a}, \beta_\mathsf{i}$$

$$O \stackrel{\mathsf{def}}{=} \bigodot_{\mathsf{i}=1}^{n} q(\mathsf{i}) \qquad n \stackrel{\mathsf{def}}{=} \#\sigma.$$

Then

$$\begin{aligned}
& Q(\sigma, \beta) \ast W(\beta, \gamma, \mathsf{a}) \\
& \Rightarrow (\#\beta{=}n \wedge \forall_{\mathsf{i}=1}^{\#\beta} p(\mathsf{i}) \ast \mathbf{true}) \ast (\#\gamma = \#\beta \wedge \bigodot_{\mathsf{i}=1}^{\#\gamma} q(\mathsf{i})) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge ((\forall_{\mathsf{i}=1}^{n} p(\mathsf{i}) \ast \mathbf{true}) \ast O) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge ((\forall \mathsf{i}.\ 1 \leq \mathsf{i} \leq n \Rightarrow p(\mathsf{i}) \ast \mathbf{true}) \ast O) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge (\forall \mathsf{i}.\ ((1 \leq \mathsf{i} \leq n \Rightarrow p(\mathsf{i}) \ast \mathbf{true}) \ast O)) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge (\forall \mathsf{i}.\ (1 \leq \mathsf{i} \leq n \Rightarrow (p(\mathsf{i}) \ast \mathbf{true} \ast O)) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge \forall_{\mathsf{i}=1}^{n} (p(\mathsf{i}) \ast \mathbf{true} \ast O) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge \forall_{\mathsf{i}=1}^{n} (p(\mathsf{i}) \ast \mathbf{true}) \wedge \\
& \qquad \forall_{\mathsf{i}=1}^{n} (p(\mathsf{i}) \ast \mathbf{true} \ast O) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge \forall_{\mathsf{i}=1}^{n} (p(\mathsf{i}) \ast \mathbf{true}) \wedge \\
& \qquad \forall_{\mathsf{i}=1}^{n} (p(\mathsf{i}) \ast \mathbf{true} \ast q(\mathsf{i})) \\
& \Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge \forall_{\mathsf{i}=1}^{n} (p(\mathsf{i}) \ast \mathbf{true}) \wedge \\
& \qquad \forall_{\mathsf{i}=1}^{n} (\mathsf{list}\,\sigma_\mathsf{i}\,\beta_\mathsf{i} \ast \mathbf{true} \ast \gamma_\mathsf{i} \mapsto \mathsf{a}, \beta_\mathsf{i}) \\
& \Rightarrow \#\gamma{=}n \wedge \#\beta{=}n \wedge \forall_{\mathsf{i}=1}^{n} (\mathsf{list}\,\sigma_\mathsf{i}\,\beta_\mathsf{i} \ast \mathbf{true}) \wedge \\
& \qquad \forall_{\mathsf{i}=1}^{n} (\mathsf{list}\,(\mathsf{ext_a}\sigma)_\mathsf{i}\,\gamma_\mathsf{i} \ast \mathbf{true}) \\
& \Rightarrow \#\gamma{\cdot}\beta = \#(\mathsf{ext_a}\sigma){\cdot}\sigma \wedge \\
& \qquad \forall_{\mathsf{i}=1}^{\#\gamma\cdot\beta} (\mathsf{list}\,((\mathsf{ext_a}\sigma){\cdot}\sigma)_\mathsf{i}\,(\gamma{\cdot}\beta)_\mathsf{i} \ast \mathbf{true}) \\
& \Rightarrow Q((\mathsf{ext_a}\sigma){\cdot}\sigma, \gamma{\cdot}\beta).
\end{aligned}$$

—

Some Details

$$\#\beta{=}n \wedge \#\gamma{=}n \wedge ((\forall \mathsf{i}.\ 1 \leq \mathsf{i} \leq n \Rightarrow p(\mathsf{i}) \ast \mathbf{true}) \ast O)$$
$$\Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge (\forall \mathsf{i}.\ ((1 \leq \mathsf{i} \leq n \Rightarrow p(\mathsf{i}) \ast \mathbf{true}) \ast O))$$

by the semidistributive law for $\ast$ and $\forall$.

$$\#\beta{=}n \wedge \#\gamma{=}n \wedge (\forall \mathsf{i}.\ ((1 \leq \mathsf{i} \leq n \Rightarrow p(\mathsf{i}) \ast \mathbf{true}) \ast O))$$
$$\Rightarrow \#\beta{=}n \wedge \#\gamma{=}n \wedge (\forall \mathsf{i}.\ (1 \leq \mathsf{i} \leq n \Rightarrow (p(\mathsf{i}) \ast \mathbf{true} \ast O))$$

since $((p \Rightarrow q) \ast r) \Rightarrow (p \Rightarrow (q \ast r))$ when p is pure.

—

Proofs (3)

$$
\begin{aligned}
& R(\beta) \, * \, W(\beta, \gamma, \mathsf{a}) \\
& \quad \Rightarrow (\beta_{\#\beta} = \mathbf{nil} \wedge \mathbf{emp}) \, * \\
& \qquad \bigodot_{\mathsf{i}=1}^{\#\gamma} \gamma_{\mathsf{i}} \mapsto \mathsf{a}, \beta_{\mathsf{i}} \, * \\
& \qquad \bigodot_{\mathsf{i}=1}^{\#\beta-1} (\exists \mathsf{a}, \mathsf{k}.\ \ \mathsf{i} < \mathsf{k} \leq \#\beta \wedge \beta_{\mathsf{i}} \mapsto \mathsf{a}, \beta_{\mathsf{k}}) \\
& \quad \Rightarrow ((\gamma \cdot \beta)_{\#\gamma \cdot \beta} = \mathbf{nil} \wedge \mathbf{emp}) \, * \\
& \qquad \bigodot_{\mathsf{i}=1}^{\#\gamma} (\exists \mathsf{a}, \mathsf{k}.\ \ \mathsf{i} \leq \#\gamma < \mathsf{k} \leq \#\gamma \cdot \beta \wedge (\gamma \cdot \beta)_{\mathsf{i}} \mapsto \mathsf{a}, (\gamma \cdot \beta)_{\mathsf{k}}) \, * \\
& \qquad \bigodot_{\mathsf{i}=\#\gamma+1}^{\#\gamma \cdot \beta-1} (\exists \mathsf{a}, \mathsf{k}.\ \ \mathsf{i} < \mathsf{k} \leq \#\gamma \cdot \beta \wedge (\gamma \cdot \beta)_{\mathsf{i}} \mapsto \mathsf{a}, (\gamma \cdot \beta)_{\mathsf{k}}) \\
& \quad \Rightarrow ((\gamma \cdot \beta)_{\#\gamma \cdot \beta} = \mathbf{nil} \wedge \mathbf{emp}) \, * \\
& \qquad \bigodot_{\mathsf{i}=1}^{\#\gamma \cdot \beta-1} (\exists \mathsf{a}, \mathsf{k}.\ \ \mathsf{i} < \mathsf{k} \leq \#\gamma \cdot \beta \wedge (\gamma \cdot \beta)_{\mathsf{i}} \mapsto \mathsf{a}, (\gamma \cdot \beta)_{\mathsf{k}}) \\
& \quad \Rightarrow R(\gamma \cdot \beta).
\end{aligned}
$$

From (2) and (3), we have

$$
\begin{aligned}
& (Q(\sigma, \beta) \wedge R(\beta)) \, * \, W(\beta, \gamma, \mathsf{a}) \\
& \qquad \Rightarrow (Q(\sigma, \beta) \, * \, W(\beta, \gamma, \mathsf{a})) \wedge (R(\beta) \, * \, W(\beta, \gamma, \mathsf{a})) \\
& \qquad \Rightarrow Q((\mathsf{ext}_{\mathsf{a}} \sigma) \cdot \sigma, \gamma \cdot \beta) \wedge R(\gamma \cdot \beta).
\end{aligned}
$$

—

A Subsidiary Recursive Procedure

$$
\begin{array}{l}
\mathsf{extapp}(\mathsf{k};\mathsf{a},\mathsf{i},\mathsf{j}) = \\
\quad \textbf{if } \mathsf{i} = \textbf{nil then } \mathsf{k} := \mathsf{j} \textbf{ else} \\
\quad\quad \textbf{newvar } \mathsf{b},\mathsf{i}',\mathsf{g} \textbf{ in} \\
\quad\quad\quad \big(\mathsf{b} := [\mathsf{i}]\ ;\ \mathsf{i}' := [\mathsf{i}+1]\ ; \\
\quad\quad\quad \mathsf{extapp}(\mathsf{k};\mathsf{a},\mathsf{i}',\mathsf{j})\ ; \\
\quad\quad\quad \mathsf{g} := \textbf{cons}(\mathsf{a},\mathsf{b})\ ;\ \mathsf{k} := \textbf{cons}(\mathsf{g},\mathsf{k})\big)
\end{array}
$$

satisfies

$$
\begin{array}{l}
\{\mathsf{list}\ \beta\ \mathsf{i}\} \\
\mathsf{extapp}(\mathsf{k};\mathsf{a},\mathsf{i},\mathsf{j})\{\beta\} \\
\{\exists\gamma.\ \ \mathsf{list}\ \beta\ \mathsf{i}\ *\ \mathsf{lseg}\ \gamma\ (\mathsf{k},\mathsf{j})\ *\ W(\beta,\gamma,\mathsf{a})\}
\end{array}
$$

—

since

$$
\begin{array}{l}
\{\mathsf{list}\ \beta\ \mathsf{i}\}\\
\textbf{if}\ \mathsf{i} = \textbf{nil then}\ \mathsf{k} := \mathsf{j}\ \textbf{else}\\
\quad \{\exists \mathsf{b}, \mathsf{i}', \beta'.\ \ \beta = \mathsf{b}{\cdot}\beta' \wedge (\mathsf{i} \mapsto \mathsf{b}, \mathsf{i}' \ * \ \mathsf{list}\ \beta'\ \mathsf{i}')\}\\
\quad \textbf{newvar}\ \mathsf{b}, \mathsf{i}', \mathsf{g}\ \textbf{in}\\
\quad\quad \big(\mathsf{b} := [\mathsf{i}]\ ;\ \mathsf{i}' := [\mathsf{i} + 1]\ ;\\
\quad\quad \{\exists \beta'.\ \ \beta = \mathsf{b}{\cdot}\beta' \wedge (\mathsf{i} \mapsto \mathsf{b}, \mathsf{i}' \ * \ \mathsf{list}\ \beta'\ \mathsf{i}')\}\\
\quad\quad \left.\begin{array}{l}
\{\mathsf{list}\ \beta'\ \mathsf{i}'\}\\
\mathsf{extapp}(\mathsf{k}; \mathsf{a}, \mathsf{i}', \mathsf{j})\{\beta'\}\\
\{\exists \gamma.\ \ \mathsf{list}\ \beta'\ \mathsf{i}' \ * \ \mathsf{lseg}\ \gamma\ (\mathsf{k}, \mathsf{j}) \ * \ W(\beta', \gamma, \mathsf{a})\}\\
\{\exists \gamma'.\ \ \mathsf{list}\ \beta'\ \mathsf{i}' * \mathsf{lseg}\ \gamma'\ (\mathsf{k}, \mathsf{j}) * W(\beta', \gamma', \mathsf{a})\}
\end{array}\right\} * \left(\begin{array}{c}\beta = \mathsf{b}{\cdot}\beta'\\ \wedge\\ \mathsf{i} \mapsto \mathsf{b}, \mathsf{i}'\end{array}\right)\Bigg\}\exists\beta\\
\quad\quad \{\exists \beta', \gamma'.\ \ \beta = \mathsf{b}{\cdot}\beta' \wedge\\
\quad\quad\quad (\mathsf{list}\ (\mathsf{b}{\cdot}\beta')\ \mathsf{i} \ * \ \mathsf{lseg}\ \gamma'\ (\mathsf{k}, \mathsf{j}) \ * \ W(\beta', \gamma', \mathsf{a}))\}\\
\quad\quad \mathsf{g} := \textbf{cons}(\mathsf{a}, \mathsf{b});\\
\quad\quad \{\exists \beta', \gamma'.\ \ \beta = \mathsf{b}{\cdot}\beta' \wedge\\
\quad\quad\quad (\mathsf{list}\ (\mathsf{b}{\cdot}\beta')\ \mathsf{i} \ * \ \mathsf{lseg}\ \gamma'\ (\mathsf{k}, \mathsf{j}) \ * \ W(\mathsf{b}{\cdot}\beta', \mathsf{g}{\cdot}\gamma', \mathsf{a}))\}\\
\quad\quad \mathsf{k} := \textbf{cons}(\mathsf{g}, \mathsf{k})\\
\quad\quad \{\exists \beta', \gamma'.\ \ \beta = \mathsf{b}{\cdot}\beta' \wedge\\
\quad\quad\quad (\mathsf{list}\ (\mathsf{b}{\cdot}\beta')\ \mathsf{i} \ * \ \mathsf{lseg}\ \mathsf{g}{\cdot}\gamma'\ (\mathsf{k}, \mathsf{j}) \ * \ W(\mathsf{b}{\cdot}\beta', \mathsf{g}{\cdot}\gamma', \mathsf{a}))\}\\
\quad\quad \big)\\
\{\exists \gamma.\ \ \mathsf{list}\ \beta\ \mathsf{i} \ * \ \mathsf{lseg}\ \gamma\ (\mathsf{k}, \mathsf{j}) \ * \ W(\beta, \gamma, \mathsf{a})\}
\end{array}
$$

—

The Main Recursive Procedure

$$
\begin{array}{l}
\mathsf{subsets}(\mathsf{j};\mathsf{i}) = \\
\quad \textbf{if } \mathsf{i} = \textbf{nil then } \mathsf{j} := \textbf{cons}(\textbf{nil}, \textbf{nil}) \textbf{ else} \\
\qquad \textbf{newvar } \mathsf{a}, \mathsf{i}', \mathsf{j}' \textbf{ in} \\
\qquad\quad \big(\mathsf{a} := [\mathsf{i}] \;;\; \mathsf{i}' := [\mathsf{i}+1] \;; \\
\qquad\quad \mathsf{subsets}(\mathsf{j}';\mathsf{i}') \;; \\
\qquad\quad \mathsf{extapp}(\mathsf{j};\mathsf{a},\mathsf{j}',\mathsf{j}')\big)
\end{array}
$$

satisfies

$$
\begin{array}{l}
\{\mathsf{list}\,\alpha\,\mathsf{i}\} \\
\mathsf{subsets}(\mathsf{j};\mathsf{i})\{\alpha\} \\
\{\exists\sigma,\beta.\ \ \mathsf{ss}(\alpha,\sigma) \wedge \big(\mathsf{list}\,\alpha\,\mathsf{i} \ * \ \mathsf{list}\,\beta\,\mathsf{j} \ * \ (Q(\sigma,\beta) \wedge R(\beta))\big)\}
\end{array}
$$

since

$$
\begin{array}{l}
\{\mathsf{list}\,\alpha\,\mathsf{i}\} \\
\textbf{if } \mathsf{i} = \textbf{nil then } \mathsf{j} := \textbf{cons}(\textbf{nil}, \textbf{nil}) \textbf{ else} \\
\qquad \vdots
\end{array}
$$

—

$$\vdots$$

$$\{\exists \mathsf{a}, \mathsf{i}', \alpha'.\ \ \alpha = \mathsf{a}{\cdot}\alpha' \wedge (\mathsf{i} \mapsto \mathsf{a}, \mathsf{i}' \ * \ \mathsf{list}\ \alpha'\, \mathsf{i}')\}$$

$$\mathbf{newvar}\ \mathsf{a}, \mathsf{i}', \mathsf{j}'\ \mathbf{in}\ \big(\mathsf{a} := [\mathsf{i}]\ ;\ \mathsf{i}' := [\mathsf{i} + 1]\ ;$$

$$\{\exists \alpha'.\ \ \alpha = \mathsf{a}{\cdot}\alpha' \wedge (\mathsf{i} \mapsto \mathsf{a}, \mathsf{i}' \ * \ \mathsf{list}\ \alpha'\, \mathsf{i}')\}$$

$$\left.\begin{array}{l}\left.\begin{array}{l}\{\mathsf{list}\ \alpha'\, \mathsf{i}'\} \\ \mathsf{subsets}(\mathsf{j}'; \mathsf{i}')\{\alpha'\} \\ \{\exists \sigma, \beta.\ \ \mathsf{ss}(\alpha', \sigma) \wedge \\ \quad (\mathsf{list}\ \alpha'\, \mathsf{i}' \ * \ \mathsf{list}\ \beta\, \mathsf{j}' \ * \ (Q(\sigma, \beta) \wedge R(\beta)))\} \\ \{\exists \sigma', \beta'.\ \ \mathsf{ss}(\alpha', \sigma') \wedge \\ \quad (\mathsf{list}\ \alpha'\, \mathsf{i}' \ * \ \mathsf{list}\ \beta'\, \mathsf{j}' \ * \ (Q(\sigma', \beta') \wedge R(\beta')))\}\end{array}\right\} \\ \hfill * \ (\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{i} \mapsto \mathsf{a}, \mathsf{i}')\end{array}\right\} \exists \alpha'$$

$$\{\exists \alpha', \sigma', \beta'.\ \ \big(\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{ss}(\alpha', \sigma') \wedge$$
$$\quad (\mathsf{list}\ (\mathsf{a}{\cdot}\alpha')\, \mathsf{i} \ * \ (Q(\sigma', \beta') \wedge R(\beta')))\big) \ * \ \mathsf{list}\ \beta'\, \mathsf{j}'\}$$

$$\left.\begin{array}{l}\left.\begin{array}{l}\{\mathsf{list}\ \beta'\, \mathsf{j}'\} \\ \mathsf{extapp}(\mathsf{j}; \mathsf{a}, \mathsf{j}', \mathsf{j}')\{\beta'\} \\ \{\exists \gamma.\ \ \mathsf{list}\ \beta'\, \mathsf{j}' \ * \ \mathsf{lseg}\ \gamma\, (\mathsf{j}, \mathsf{j}') \ * \ W(\beta', \gamma, \mathsf{a})\}\end{array}\right\} \\ \qquad * \ \big(\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{ss}(\alpha', \sigma') \wedge \\ \qquad\quad (\mathsf{list}\ (\mathsf{a}{\cdot}\alpha')\, \mathsf{i} \ * \ (Q(\sigma', \beta') \wedge R(\beta')))\big)\end{array}\right\} \exists \alpha', \sigma', \beta'$$

$$\{\exists \alpha', \sigma', \beta'.\ \ \big(\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{ss}(\alpha', \sigma') \wedge$$
$$\qquad (\mathsf{list}\ (\mathsf{a}{\cdot}\alpha')\, \mathsf{i} \ * \ (Q(\sigma', \beta') \wedge R(\beta')))\big) \ *$$
$$\quad (\exists \gamma.\ \ \mathsf{list}\ \beta'\, \mathsf{j}' \ * \ \mathsf{lseg}\ \gamma\, (\mathsf{j}, \mathsf{j}') \ * \ W(\beta', \gamma, \mathsf{a}))\}$$

$$\{\exists \alpha', \sigma', \beta', \gamma.\ \ \alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{ss}(\mathsf{a}{\cdot}\alpha', (\mathsf{ext}_{\mathsf{a}}\sigma'){\cdot}\sigma') \wedge$$
$$\quad \big(\mathsf{list}\ (\mathsf{a}{\cdot}\alpha')\, \mathsf{i} \ * \ \mathsf{list}\ (\gamma{\cdot}\beta')\, \mathsf{j} \ *$$
$$\qquad (Q((\mathsf{ext}_{\mathsf{a}}\sigma'){\cdot}\sigma', \gamma{\cdot}\beta') \wedge R(\gamma{\cdot}\beta'))\big)\}\big)$$

$$\{\exists \sigma, \beta.\ \ \mathsf{ss}(\alpha, \sigma) \wedge \big(\mathsf{list}\ \alpha\, \mathsf{i} \ * \ \mathsf{list}\ \beta\, \mathsf{j} \ * \ (Q(\sigma, \beta) \wedge R(\beta))\big)\}$$

—

Exercise 1

Derive the axiom scheme

$$m \leq j \leq n \Rightarrow \Big(\Big(\bigodot_{i=m}^{n} p(i)\Big) \Rightarrow (p(j) \ * \ \mathbf{true})\Big)$$

from the other axiom schemata for iterating separating conjunction.

—

Exercise 2

The following is an alternative global rule for allocation that uses a ghost variable (v'):

- The ghost-variable global form (ALLOCGG)

  $$\frac{}{\{v = v' \land r\}\ v := \mathbf{allocate}\ e\ \{(\bigodot_{i=v}^{v+e'-1} i \mapsto -)\ *\ r'\},}$$

  where v' is distinct from v, e' denotes $e/v \to v'$, and r' denotes $r/v \to v'$.

Derive (ALLOCGG) from (ALLOCG) and (ALLOCL) from (ALLOCGG).

—

Exercise 3

Write an iterative version (in which recursion or, for that matter, procedures are not used) of the program for subset lists in the class notes. Since it is natural for efficent iterative programs to reverse lists, your program will not give exactly the same results as the one in the notes.

Specifically, you will need to replace the predicates ss and W by

$$\mathsf{ss}'(\epsilon, \sigma) \stackrel{\mathrm{def}}{=} \sigma = [\epsilon]$$
$$\mathsf{ss}'(\mathsf{a}\cdot\alpha, \sigma) \stackrel{\mathrm{def}}{=} \exists \sigma'. \left(\mathsf{ss}'(\alpha, \sigma') \wedge \sigma = (\mathsf{ext}_\mathsf{a}\sigma')^\dagger \cdot \sigma'\right)$$

and

$$W'(\beta, \gamma, \mathsf{a}) \stackrel{\mathrm{def}}{=} \#\gamma = \#\beta \wedge \bigodot_{\mathsf{i}=1}^{\#\gamma} \gamma_\mathsf{i} \mapsto \mathsf{a}, (\beta^\dagger)_\mathsf{i}.$$

—

Then your program should contain a nest of two **while** commands. It should satisfy

$$\{\mathsf{list}\ \alpha\ \mathsf{i}\}$$
"Set j to list of lists of subsets of i"
$$\{\exists \sigma, \beta.\ \ \mathsf{ss}'(\alpha^\dagger, \sigma) \wedge (\mathsf{list}\ \beta\ \mathsf{j}\ *\ (Q(\sigma, \beta) \wedge R(\beta)))\}.$$

The invariant of the outer **while** should be

$$\exists \alpha', \alpha'', \sigma, \beta.\ \ \alpha'^\dagger{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}'(\alpha', \sigma) \wedge$$
$$(\mathsf{list}\ \alpha''\ \mathsf{i}\ *\ \mathsf{list}\ \beta\ \mathsf{j}\ *\ (Q(\sigma, \beta) \wedge R(\beta))),$$

and the invariant of the inner **while** should be

$$\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma.\ \ \alpha'^\dagger{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}'(\alpha', \sigma) \wedge$$
$$(\mathsf{list}\ \alpha''\ \mathsf{i}\ *\ \mathsf{lseg}\ \gamma\ (\mathsf{l}, \mathsf{j})\ *\ \mathsf{lseg}\ \beta'\ (\mathsf{j}, \mathsf{m})\ *\ \mathsf{list}\ \beta''\ \mathsf{m}\ *$$
$$(Q(\sigma, \beta'{\cdot}\beta'') \wedge R(\beta'{\cdot}\beta''))\ *\ W'(\beta', \gamma, \mathsf{a})).$$

At the completion of the inner **while**, the assertion

$$\exists \alpha', \alpha'', \sigma, \beta', \gamma.\ \ \alpha'^\dagger{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}'(\alpha', \sigma) \wedge$$
$$(\mathsf{list}\ \alpha''\ \mathsf{i}\ *\ \mathsf{lseg}\ \gamma\ (\mathsf{l}, \mathsf{j})\ *\ \mathsf{list}\ \beta'\ \mathsf{j}\ *$$
$$(Q(\sigma, \beta') \wedge R(\beta'))\ *\ W'(\beta', \gamma, \mathsf{a}))$$

should hold.

—