

CLASS NOTES FOR CS 818A3 - SPRING 2005

AN INTRODUCTION TO SEPARATION LOGIC

5. Iterated Separating Conjunction

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

February 18, 2005

©2005 John C. Reynolds

Iterated Separating Conjunction

$$\langle \text{assert} \rangle ::= \dots \mid \odot_{\langle \text{var} \rangle = \langle \text{exp} \rangle}^{\langle \text{exp} \rangle} \langle \text{assert} \rangle$$

$$s, h \models \odot_{v=e}^{e'} p \text{ iff}$$

$$\text{let } m = \llbracket e \rrbracket_{\text{exp}} s, \quad n = \llbracket e' \rrbracket_{\text{exp}} s, \quad I = \{ i \mid m \leq i \leq n \} \text{ in}$$

$$\exists H \in I \rightarrow \text{Heaps.}$$

$$\forall i, j \in I. i \neq j \text{ implies } Hi \perp Hj$$

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

$$\text{and } \forall i \in I. [s \mid v:i], Hi \models p.$$

Axiom Schemata

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

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

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

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

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

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

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

Array Allocation

$$\langle \text{comm} \rangle ::= \dots \mid \langle \text{var} \rangle := \mathbf{allocate} \langle \text{exp} \rangle$$

Semantics

$$\frac{}{(s, h) \llbracket v := \mathbf{allocate} e \rrbracket_{\text{comm}} ([s \mid v: \ell], h \cdot h'),}$$

where $h \perp h'$ and $\text{dom } h' = \{i \mid \ell \leq i < \ell + \llbracket e \rrbracket_{\text{exp}} s\}$.

Inference Rules

Noninterfering:

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

where v does not occur free in r or e .

The local form:

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

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

The global form:

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

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

The backward-reasoning form:

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

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

A Cyclic Buffer Using an Array

We assume that an n -element array has been allocated at location l . Let $\phi(x)$ be the unique integer such that

$$\phi(x) \equiv_{\text{mod } n} x \quad l \leq \phi(x) < l + n.$$

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

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

It is easy to show that

$$\{R \wedge m < n\}$$

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

$$\{R\}$$

and

$$\{R \wedge m > 0\}$$

$$m := m - 1 ; \text{if } i = l + n - 1 \text{ then } i := l \text{ else } i := i + 1$$

$$\{R\}$$

When the buffer contains a sequence α , it should satisfy

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

Inserting an Element

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

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

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

[j] := x ;

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

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

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

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

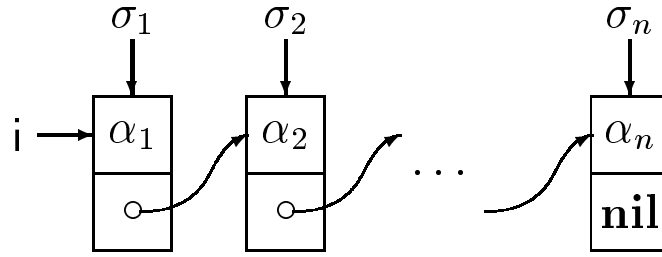
$$\{((\odot_{k=0}^m \phi(i+k) \mapsto (\alpha \cdot x)_{k+1}) * (\odot_{k=1}^{n-m-1} \phi(j+k) \mapsto -)) \\ \wedge m + 1 = \#(\alpha \cdot x) \wedge R \wedge m < n\}$$

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

m := m + 1 ; **if** j = l + n - 1 **then** j := l **else** j := j + 1

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

Connecting Two Views of Lists



If

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

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

and

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

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

then

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

The proof is by induction on α .

Specifying Subset Lists

Let $\alpha, \beta \in \mathbf{Z}^*$ and $\sigma \in (\mathbf{Z}^*)^*$ in

{list α i}

“Set j to list of lists of subsets of i”

{ $\exists \sigma, \beta. \text{ss}(\alpha^\dagger, \sigma) \wedge (\text{list } \beta \text{ j} * (Q(\sigma, \beta) \wedge R(\beta)))$ },

where

$$\# \text{ext}_a \sigma \stackrel{\text{def}}{=} \# \sigma$$

$$(\text{ext}_a \sigma)_i \stackrel{\text{def}}{=} a \cdot \sigma_i$$

$$\text{ss}(\epsilon, \sigma) \stackrel{\text{def}}{=} \sigma = [\epsilon]$$

$$\text{ss}(a \cdot \alpha, \sigma) \stackrel{\text{def}}{=} \exists \sigma'. \text{ss}(\alpha, \sigma') \wedge \sigma = (\text{ext}_a \sigma')^\dagger \cdot \sigma'$$

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

$$R(\beta) \stackrel{\text{def}}{=} (\beta_{\# \beta} = \mathbf{nil} \wedge \mathbf{emp}) *$$

$$\odot_{i=1}^{\# \beta - 1} (\exists a, k. i < k \leq \# \beta \wedge \beta_i \mapsto a, \beta_k).$$

Some Properties

The predicates

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

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

$$W(\beta, \gamma, \mathbf{a}) \stackrel{\text{def}}{=} \# \gamma = \# \beta \wedge \odot_{i=1}^{\# \gamma} \gamma_i \mapsto \mathbf{a}, (\beta^\dagger)_i$$

satisfy

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

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

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

$$Q(\sigma, \beta) * W(\beta, \gamma, \mathbf{a}) \Rightarrow Q((\text{ext}_{\mathbf{a}} \sigma)^\dagger \cdot \sigma, \gamma \cdot \beta)$$

$$R(\beta) * W(\beta, \gamma, \mathbf{a}) \Rightarrow R(\gamma \cdot \beta)$$

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

Proofs (1)

$$W(\beta, \gamma, \mathbf{a}) * \mathbf{g} \mapsto \mathbf{a}, \mathbf{b}$$

$$\Leftrightarrow \#\gamma = \#\beta \wedge (\mathbf{g} \mapsto \mathbf{a}, \mathbf{b} * \odot_{i=1}^{\#\gamma} \gamma_i \mapsto \mathbf{a}, (\beta^\dagger)_i)$$

$$\Leftrightarrow \#\mathbf{g} \cdot \gamma = \#\beta \cdot \mathbf{b} \wedge \left(\left(\odot_{i=1}^1 (\mathbf{g} \cdot \gamma)_i \mapsto \mathbf{a}, ((\beta \cdot \mathbf{b})^\dagger)_i \right) * \right. \\ \left. \left(\odot_{i=1}^{\#\mathbf{g} \cdot \gamma - 1} (\mathbf{g} \cdot \gamma)_{i+1} \mapsto \mathbf{a}, ((\beta \cdot \mathbf{b})^\dagger)_{i+1} \right) \right)$$

$$\Leftrightarrow \#\mathbf{g} \cdot \gamma = \#\beta \cdot \mathbf{b} \wedge \odot_{i=1}^{\#\mathbf{g} \cdot \gamma} (\mathbf{g} \cdot \gamma)_i \mapsto \mathbf{a}, ((\beta \cdot \mathbf{b})^\dagger)_i$$

$$\Leftrightarrow W(\beta \cdot \mathbf{b}, \mathbf{g} \cdot \gamma, \mathbf{a}).$$

Proofs (2)

Let

$$p(i) \stackrel{\text{def}}{=} \text{list } \sigma_i \beta_i$$

$$q(i) \stackrel{\text{def}}{=} \gamma_i \mapsto \mathbf{a}, (\beta^\dagger)_i$$

$$n \stackrel{\text{def}}{=} \#\sigma.$$

Then

$$Q(\sigma, \beta) * W(\beta, \gamma, \mathbf{a})$$

$$\Rightarrow \left(\#\beta = n \wedge \forall_{i=1}^{\#\beta} (p(i) * \mathbf{true}) \right) * \left(\#\gamma = \#\beta \wedge \odot_{i=1}^{\#\gamma} q(i) \right)$$

$$\Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \left(\left(\forall_{i=1}^n (p(i) * \mathbf{true}) \right) * \odot_{i=1}^n q(i) \right)$$

$$\Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{i=1}^n \left(p(i) * \mathbf{true} * \odot_{j=1}^n q(j) \right)$$

$$\Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{i=1}^n \left(p(i) * \mathbf{true} \right) \wedge$$

$$\forall_{i=1}^n \left(p(n+1-i) * \mathbf{true} * \odot_{j=1}^n q(j) \right)$$

$$\Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{i=1}^n \left(p(i) * \mathbf{true} \right) \wedge$$

$$\forall_{i=1}^n \left(p(n+1-i) * \mathbf{true} * q(i) \right)$$

$$\Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{i=1}^n \left(p(i) * \mathbf{true} \right) \wedge$$

$$\forall_{i=1}^n \left(\text{list } (\sigma^\dagger)_i (\beta^\dagger)_i * \mathbf{true} * \gamma_i \mapsto \mathbf{a}, (\beta^\dagger)_i \right)$$

$$\Rightarrow \#\gamma = n \wedge \#\beta = n \wedge \forall_{i=1}^n \left(\text{list } \sigma_i \beta_i * \mathbf{true} \right) \wedge$$

$$\forall_{i=1}^n \left(\text{list } ((\text{ext}_{\mathbf{a}}\sigma)^\dagger)_i \gamma_i * \mathbf{true} \right)$$

$$\Rightarrow \#\gamma \cdot \beta = \#(\text{ext}_{\mathbf{a}}\sigma)^\dagger \cdot \sigma \wedge$$

$$\forall_{i=1}^{\#\gamma \cdot \beta} \left(\text{list } ((\text{ext}_{\mathbf{a}}\sigma)^\dagger \cdot \sigma)_i (\gamma \cdot \beta)_i * \mathbf{true} \right)$$

$$\Rightarrow Q((\text{ext}_{\mathbf{a}}\sigma)^\dagger \cdot \sigma, \gamma \cdot \beta).$$

Proofs (3)

$$R(\beta) * W(\beta, \gamma, \mathbf{a})$$

$$\Rightarrow (\beta_{\#\beta} = \mathbf{nil} \wedge \mathbf{emp}) *$$

$$\odot_{i=1}^{\#\gamma} \gamma_i \mapsto \mathbf{a}, (\beta^\dagger)_i *$$

$$\odot_{i=1}^{\#\beta-1} (\exists \mathbf{a}, \mathbf{k}. i < \mathbf{k} \leq \#\beta \wedge \beta_i \mapsto \mathbf{a}, \beta_{\mathbf{k}})$$

$$\Rightarrow ((\gamma \cdot \beta)_{\#\gamma \cdot \beta} = \mathbf{nil} \wedge \mathbf{emp}) *$$

$$\odot_{i=1}^{\#\gamma} (\exists \mathbf{a}, \mathbf{k}. \#\gamma < \mathbf{k} \leq \#\gamma \cdot \beta \wedge (\gamma \cdot \beta)_i \mapsto \mathbf{a}, (\gamma \cdot \beta)_{\mathbf{k}}) *$$

$$\odot_{i=\#\gamma+1}^{\#\gamma \cdot \beta-1} (\exists \mathbf{a}, \mathbf{k}. i < \mathbf{k} \leq \#\gamma \cdot \beta \wedge (\gamma \cdot \beta)_i \mapsto \mathbf{a}, (\gamma \cdot \beta)_{\mathbf{k}})$$

$$\Rightarrow ((\gamma \cdot \beta)_{\#\gamma \cdot \beta} = \mathbf{nil} \wedge \mathbf{emp}) *$$

$$\odot_{i=1}^{\#\gamma \cdot \beta-1} (\exists \mathbf{a}, \mathbf{k}. i < \mathbf{k} \leq \#\gamma \cdot \beta \wedge (\gamma \cdot \beta)_i \mapsto \mathbf{a}, (\gamma \cdot \beta)_{\mathbf{k}})$$

$$\Rightarrow R(\gamma \cdot \beta).$$

From (2) and (3), we have

$$(Q(\sigma, \beta) \wedge R(\beta)) * W(\beta, \gamma, \mathbf{a})$$

$$\Rightarrow (Q(\sigma, \beta) * W(\beta, \gamma, \mathbf{a})) \wedge (R(\beta) * W(\beta, \gamma, \mathbf{a}))$$

$$\Rightarrow Q((\text{ext}_{\mathbf{a}}\sigma)^\dagger \cdot \sigma, \gamma \cdot \beta) \wedge R(\gamma \cdot \beta).$$

An Iterative Program for Subset Lists

Final Assertion

$$\exists \sigma, \beta. \text{ss}(\alpha^\dagger, \sigma) \wedge (\text{list } \beta \text{ j} * (Q(\sigma, \beta) \wedge R(\beta)))$$

Outer Invariant

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

Inner Invariant

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

After inner loop:

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

Iterative Subset Lists: The Main Program

$\{\text{list } \alpha \ i\}$

$j := \mathbf{cons}(\mathbf{nil}, \mathbf{nil}) ;$

$\{\text{list } \alpha \ i * j \mapsto \mathbf{nil}, \mathbf{nil}\}$

$\{\text{list } \alpha \ i * \text{list } [\mathbf{nil}] \ j\}$

$\{\mathbf{ss}(\epsilon, [\epsilon]) \wedge (\text{list } \alpha \ i * \text{list } [\mathbf{nil}] \ j * (Q([\epsilon], [\mathbf{nil}]) \wedge R([\mathbf{nil}])))\}$

$\{\exists \alpha', \alpha'', \sigma, \beta. \alpha'^{\dagger} \cdot \alpha'' = \alpha \wedge \mathbf{ss}(\alpha', \sigma) \wedge$

$(\text{list } \alpha'' \ i * \text{list } \beta \ j * (Q(\sigma, \beta) \wedge R(\beta)))\}$

while $i \neq \mathbf{nil}$ **do** *Body of the Outer Loop*

$\{\exists \sigma, \beta. \mathbf{ss}(\alpha^{\dagger}, \sigma) \wedge (\text{list } \beta \ j * (Q(\sigma, \beta) \wedge R(\beta)))\}$

Iterative Subset Lists: Body of the Outer Loop

$$\{ \exists \alpha', \alpha'', \sigma, \beta, \mathbf{a}, \mathbf{k}. \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(i \mapsto \mathbf{a}, \mathbf{k} * \text{list } \alpha'' \mathbf{k} * \text{list } \beta \mathbf{j} * (Q(\sigma, \beta) \wedge R(\beta))) \}$$

$$(\mathbf{a} := [i]; \mathbf{k} := [i + 1]; \text{dispose } i; \text{dispose } i + 1; i := \mathbf{k};$$

$$\{ \exists \alpha', \alpha'', \sigma, \beta. \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' i * \text{list } \beta \mathbf{j} * (Q(\sigma, \beta) \wedge R(\beta))) \}$$

$$\{ \exists \alpha', \alpha'', \sigma, \beta. \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' i * \text{lseg } \epsilon (j, j) * \text{lseg } \epsilon (j, j) * \text{list } \beta \mathbf{j} *$$

$$(Q(\sigma, \epsilon \cdot \beta) \wedge R(\epsilon \cdot \beta)) * W(\epsilon, \epsilon, \mathbf{a})) \}$$

$$l := j; m := j;$$

$$\{ \exists \alpha', \alpha'', \sigma, \beta. \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' i * \text{lseg } \epsilon (l, j) * \text{lseg } \epsilon (j, m) * \text{list } \beta \mathbf{m} *$$

$$(Q(\sigma, \epsilon \cdot \beta) \wedge R(\epsilon \cdot \beta)) * W(\epsilon, \epsilon, \mathbf{a})) \}$$

$$\{ \exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma. \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' i * \text{lseg } \gamma (l, j) * \text{lseg } \beta' (j, m) * \text{list } \beta'' \mathbf{m} *$$

$$(Q(\sigma, \beta' \cdot \beta'') \wedge R(\beta' \cdot \beta'')) * W(\beta', \gamma, \mathbf{a})) \}$$

while $\mathbf{m} \neq \mathbf{nil}$ **do** *Body of the Inner Loop*

$$\{ \exists \alpha', \alpha'', \sigma, \beta', \gamma. \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' i * \text{lseg } \gamma (l, j) * \text{list } \beta' \mathbf{j} *$$

$$(Q(\sigma, \beta') \wedge R(\beta')) * W(\beta', \gamma, \mathbf{a})) \}$$

$$\{ \exists \alpha', \alpha'', \sigma, \beta', \gamma. (\mathbf{a} \cdot \alpha')^{\dagger} \cdot \alpha'' = \alpha \wedge \text{ss}(\mathbf{a} \cdot \alpha', (\text{ext}_{\mathbf{a}} \sigma)^{\dagger} \cdot \sigma) \wedge$$

$$(\text{list } \alpha'' i * \text{list } (\gamma \cdot \beta') l * (Q((\text{ext}_{\mathbf{a}} \sigma)^{\dagger} \cdot \sigma, \gamma \cdot \beta') \wedge R(\gamma \cdot \beta'))) \}$$

$$\{ \exists \alpha', \alpha'', \sigma, \beta. \alpha'^{\dagger} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' i * \text{list } \beta l * (Q(\sigma, \beta) \wedge R(\beta))) \}$$

$$j := l)$$

$$\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathbf{b}, \mathbf{m}'' . \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' \mathbf{i} * \text{lseg } \gamma (l, j) * \text{lseg } \beta' (j, m) * m \mapsto \mathbf{b}, \mathbf{m}'' * \\ \text{list } \beta'' \mathbf{m}'' * (Q(\sigma, \beta' \cdot \mathbf{b} \cdot \beta'') \wedge R(\beta' \cdot \mathbf{b} \cdot \beta'')) * W(\beta', \gamma, \mathbf{a}))\}$$

$$(\mathbf{b} := [\mathbf{m}] ;$$

$$\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathbf{m}'' . \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' \mathbf{i} * \text{lseg } \gamma (l, j) * \text{lseg } \beta' (j, m) * m \mapsto \mathbf{b}, \mathbf{m}'' * \\ \text{list } \beta'' \mathbf{m}'' * (Q(\sigma, \beta' \cdot \mathbf{b} \cdot \beta'') \wedge R(\beta' \cdot \mathbf{b} \cdot \beta'')) * W(\beta', \gamma, \mathbf{a}))\}$$

$$\mathbf{m} := [\mathbf{m} + 1] ;$$

$$\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathbf{m}' . \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' \mathbf{i} * \text{lseg } \gamma (l, j) * \text{lseg } \beta' (j, \mathbf{m}') * \mathbf{m}' \mapsto \mathbf{b}, \mathbf{m} * \\ \text{list } \beta'' \mathbf{m} * (Q(\sigma, \beta' \cdot \mathbf{b} \cdot \beta'') \wedge R(\beta' \cdot \mathbf{b} \cdot \beta'')) * W(\beta', \gamma, \mathbf{a}))\}$$

$$\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma . \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' \mathbf{i} * \text{lseg } \gamma (l, j) * \text{lseg } \beta' \cdot \mathbf{b} (j, m) * \text{list } \beta'' \mathbf{m} * \\ (Q(\sigma, \beta' \cdot \mathbf{b} \cdot \beta'') \wedge R(\beta' \cdot \mathbf{b} \cdot \beta'')) * W(\beta', \gamma, \mathbf{a}))\}$$

$$\mathbf{g} := \mathbf{cons}(\mathbf{a}, \mathbf{b}) ;$$

$$\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma . \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' \mathbf{i} * \text{lseg } \gamma (l, j) * \text{lseg } \beta' \cdot \mathbf{b} (j, m) * \text{list } \beta'' \mathbf{m} * \\ (Q(\sigma, \beta' \cdot \mathbf{b} \cdot \beta'') \wedge R(\beta' \cdot \mathbf{b} \cdot \beta'')) * W(\beta', \gamma, \mathbf{a}) * \mathbf{g} \mapsto \mathbf{a}, \mathbf{b})\}$$

$$\mathbf{l} := \mathbf{cons}(\mathbf{g}, \mathbf{l})$$

$$\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathbf{l}' . \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' \mathbf{i} * \mathbf{l} \mapsto \mathbf{g}, \mathbf{l}' * \text{lseg } \gamma (\mathbf{l}', j) * \text{lseg } \beta' \cdot \mathbf{b} (j, m) * \text{list } \beta'' \mathbf{m} * \\ (Q(\sigma, \beta' \cdot \mathbf{b} \cdot \beta'') \wedge R(\beta' \cdot \mathbf{b} \cdot \beta'')) * W(\beta', \gamma, \mathbf{a}) * \mathbf{g} \mapsto \mathbf{a}, \mathbf{b})\}$$

$$\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma . \alpha'^{\dagger} \cdot \mathbf{a} \cdot \alpha'' = \alpha \wedge \text{ss}(\alpha', \sigma) \wedge$$

$$(\text{list } \alpha'' \mathbf{i} * \text{lseg } \mathbf{g} \cdot \gamma (l, j) * \text{lseg } \beta' \cdot \mathbf{b} (j, m) * \text{list } \beta'' \mathbf{m} * \\ (Q(\sigma, \beta' \cdot \mathbf{b} \cdot \beta'') \wedge R(\beta' \cdot \mathbf{b} \cdot \beta'')) * W(\beta' \cdot \mathbf{b}, \mathbf{g} \cdot \gamma, \mathbf{a}))\} ;$$

A Recursive Program for Subset Lists

Let

$$W'(\beta, \gamma, a) \stackrel{\text{def}}{=} \#\gamma = \#\beta \wedge \bigodot_{i=1}^{\#\gamma} \gamma_i \mapsto a, \beta_i,$$

which satisfies

$$W'(\beta, \gamma, a) * g \mapsto a, b \Leftrightarrow W'(b \cdot \beta, g \cdot \gamma, a).$$

Then the procedure

```

extapp(a, i, j; k) =
  if i = nil then k := j else
    newvar b, i', g in
      ( b := [i] ; i' := [i + 1] ;
        extapp(a, i', j; k) ;
        g := cons(a, b) ; k := cons(g, k) )

```

satisfies

```

{list β i}
extapp(a, i, j; k)
{∃γ. list β i * lseg γ (k, j) * W'(β, γ, a)}

```

since

$\{\text{list } \beta \text{ } i\}$

if $i = \text{nil}$ **then** $k := j$ **else**

$\{\exists b, i', \beta'. \beta = b \cdot \beta' \wedge (i \mapsto b, i' * \text{list } \beta' \text{ } i')\}$

newvar b, i', g **in**

$(b := [i]; i' := [i + 1];$

$\{\exists \beta'. \beta = b \cdot \beta' \wedge (i \mapsto b, i' * \text{list } \beta' \text{ } i')\}$

$\{\exists \beta'. (\beta = b \cdot \beta' \wedge i \mapsto b, i') * \text{list } \beta' \text{ } i'\}$

$\text{extapp}(a, i', j; k);$

$\{\exists \beta'. (\beta = b \cdot \beta' \wedge i \mapsto b, i') *$

$(\exists \gamma'. \text{list } \beta' \text{ } i' * \text{lseg } \gamma' (k, j) * W'(\beta', \gamma', a))\}$

$\{\exists \beta', \gamma'. \beta = b \cdot \beta' \wedge$

$(\text{list } (b \cdot \beta') \text{ } i * \text{lseg } \gamma' (k, j) * W'(\beta', \gamma', a))\}$

$g := \text{cons}(a, b);$

$\{\exists \beta', \gamma'. \beta = b \cdot \beta' \wedge$

$(\text{list } (b \cdot \beta') \text{ } i * \text{lseg } \gamma' (k, j) * W'(b \cdot \beta', g \cdot \gamma', a))\}$

$k := \text{cons}(g, k)$

$\{\exists \beta', \gamma'. \beta = b \cdot \beta' \wedge$

$(\text{list } (b \cdot \beta') \text{ } i * \text{lseg } g \cdot \gamma' (k, j) * W'(b \cdot \beta', g \cdot \gamma', a))\}$

)

$\{\exists \gamma. \text{list } \beta \text{ } i * \text{lseg } \gamma (k, j) * W'(\beta, \gamma, a)\}$

Reasoning about the Call

From the recursion assumption:

$$\begin{aligned} & \{\text{list } \beta \ i\} \\ & \text{extapp}(a, i, j; k) \\ & \{\exists \gamma. \text{list } \beta \ i * \text{lseg } \gamma \ (k, j) * W'(\beta, \gamma, a)\} \end{aligned}$$

by free variable substitution (FVS):

$$\begin{aligned} & \{\text{list } \beta' \ i'\} \\ & \text{extapp}(a, i', j; k) \\ & \{\exists \gamma. \text{list } \beta' \ i' * \text{lseg } \gamma \ (k, j) * W'(\beta', \gamma, a)\} \end{aligned}$$

by renaming (RN) in the postcondition:

$$\begin{aligned} & \{\text{list } \beta' \ i'\} \\ & \text{extapp}(a, i', j; k) \\ & \{\exists \gamma'. \text{list } \beta' \ i' * \text{lseg } \gamma' \ (k, j) * W'(\beta', \gamma', a)\} \end{aligned}$$

by the frame rule (FR):

$$\begin{aligned} & \{(\beta = b \cdot \beta' \wedge i \mapsto b, i') * \text{list } \beta' \ i'\} \\ & \text{extapp}(a, i', j; k) ; \\ & \{(\beta = b \cdot \beta' \wedge i \mapsto b, i') * \\ & \quad (\exists \gamma'. \text{list } \beta' \ i' * \text{lseg } \gamma' \ (k, j) * W'(\beta', \gamma', a))\} \end{aligned}$$

by introducing existential quantification (EQ):

$$\begin{aligned} & \{\exists \beta'. (\beta = b \cdot \beta' \wedge i \mapsto b, i') * \text{list } \beta' \ i'\} \\ & \text{extapp}(a, i', j; k) ; \\ & \{\exists \beta'. (\beta = b \cdot \beta' \wedge i \mapsto b, i') * \\ & \quad (\exists \gamma'. \text{list } \beta' \ i' * \text{lseg } \gamma' \ (k, j) * W'(\beta', \gamma', a))\} \end{aligned}$$

Another Recursive Procedure

Let

$$ss'(\epsilon, \sigma) \stackrel{\text{def}}{=} \sigma = [\epsilon]$$

$$ss'(\mathbf{a} \cdot \alpha, \sigma) \stackrel{\text{def}}{=} \exists \sigma'. ss'(\alpha, \sigma') \wedge \sigma = (\text{ext}_a \sigma') \cdot \sigma'$$

and, as before,

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

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

$$\# \text{ext}_a \sigma \stackrel{\text{def}}{=} \# \sigma$$

$$(\text{ext}_a \sigma)_i \stackrel{\text{def}}{=} \mathbf{a} \cdot \sigma_i,$$

which satisfy

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

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

$$Q(\sigma, \beta) * W'(\beta, \gamma, \mathbf{a}) \Rightarrow Q((\text{ext}_a \sigma) \cdot \sigma, \gamma \cdot \beta)$$

$$R(\beta) * W'(\beta, \gamma, \mathbf{a}) \Rightarrow R(\gamma \cdot \beta)$$

$$(Q(\sigma, \beta) \wedge R(\beta)) * W'(\beta, \gamma, \mathbf{a}) \Rightarrow Q((\text{ext}_a \sigma) \cdot \sigma, \gamma \cdot \beta) \wedge R(\gamma \cdot \beta).$$

Then the procedure

```

subsets(i; j) =
  if i = nil then j := cons(nil, nil) else
    newvar a, i', j' in
      (a := [i] ; i' := [i + 1] ;
       subsets(i'; j') ;
       extapp(a, j', j'; j))

```

satisfies

$$\begin{aligned}
 & \{\text{list } \alpha \text{ } i\} \\
 & \text{subsets}(i; j) \\
 & \{\exists \sigma, \beta. \text{ss}'(\alpha, \sigma) \wedge (\text{list } \alpha \text{ } i * \text{list } \beta \text{ } j * (Q(\sigma, \beta) \wedge R(\beta)))\}
 \end{aligned}$$

since

$\{\text{list } \alpha \text{ } i\}$

if $i = \text{nil}$ **then** $j := \text{cons}(\text{nil}, \text{nil})$ **else**

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

newvar a, i', j' **in**

$(a := [i] ; i' := [i + 1] ;$

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

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

$\text{subsets}(i'; j') ;$

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

$(\exists \sigma', \beta'. \text{ss}'(\alpha', \sigma') \wedge (\text{list } \alpha' i' *$

$\text{list } \beta' j' * (Q(\sigma', \beta') \wedge R(\beta'))))\}$

$\{\exists \alpha', \sigma', \beta'. (\alpha = a \cdot \alpha' \wedge \text{ss}'(\alpha', \sigma') \wedge$

$(\text{list } (a \cdot \alpha') i * (Q(\sigma', \beta') \wedge R(\beta')))) *$

$\text{list } \beta' j'\}$

$\text{extapp}(a, j', j'; j)$

$\{\exists \alpha', \sigma', \beta'. (\alpha = a \cdot \alpha' \wedge \text{ss}'(\alpha', \sigma') \wedge$

$(\text{list } (a \cdot \alpha') i * (Q(\sigma', \beta') \wedge R(\beta')))) *$

$(\exists \gamma. \text{list } \beta' j' * \text{lseg } \gamma(j, j') * W'(\beta', \gamma, a))\}$

$\{\exists \alpha', \sigma', \beta', \gamma. \alpha = a \cdot \alpha' \wedge \text{ss}'(a \cdot \alpha', (\text{ext}_a \sigma') \cdot \sigma') \wedge$

$(\text{list } (a \cdot \alpha') i * \text{list } (\gamma \cdot \beta') j *$

$(Q((\text{ext}_a \sigma') \cdot \sigma', \gamma \cdot \beta') \wedge R(\gamma \cdot \beta')))\}$

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

Arrays that Denote Sequences

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

Properties

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

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

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

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

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

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

$$\text{array } \alpha (a, c) * \text{array } \beta (c + 1, b)$$

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

$$\Leftrightarrow \text{array } \alpha \cdot \beta (a, b) \wedge c = b - \# \beta$$

A Subtlety

In the following annotated specification, note that the assertion following the second **else** depends upon

$$c + 1 \hookrightarrow x \wedge d - 1 \hookrightarrow y \wedge x > r \wedge y \leq r \Rightarrow c + 1 \neq d - 1.$$

Partition

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

newvar d, x, y **in** $(c := a - 1 ; d := b + 1 ;$

$\{\exists \alpha_1, \alpha_2, \alpha_3. (\text{array } \alpha_1(a, c) * \text{array } \alpha_2(c + 1, d - 1) * \text{array } \alpha_3(d, b))$

$\wedge \alpha_1 \cdot \alpha_2 \cdot \alpha_3 \sim \alpha \wedge \{\alpha_1\} \leq^* r \wedge \{\alpha_3\} >^* r\}$

while $d > c + 1$ **do** $(x := [c + 1];$

if $x \leq r$ **then**

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

$c := c + 1$

else $(y := [d - 1];$

if $y > r$ **then**

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

$d := d - 1$

else

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

$([c + 1] := y ; [d - 1] := x ; c := c + 1 ; d := d - 1))$

$\{\exists \alpha_1, \alpha_2. (\text{array } \alpha_1(a, c) * \text{array } \alpha_2(c + 1, b))$

$\wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* r \wedge r <^* \{\alpha_2\}\}$

From Partition to Quicksort

Thus, if we define

```

partition(a, b, r; c) =
  newvar d, x, y in (c := a - 1 ; d := b + 1 ;
  while d > c + 1 do
    (x := [c + 1] ; if x ≤ r then c := c + 1 else
      (y := [d - 1] ; if y > r then d := d - 1 else
        ([c + 1] := y ; [d - 1] := x ; c := c + 1 ; d := d - 1))))),

```

we have

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

Now assume

$$\begin{aligned}
 & \{\text{array } \alpha(a, b)\} \\
 & \text{quicksort}(a, b) \\
 & \{\exists \beta. \text{array } \beta(a, b) \wedge \beta \sim \alpha \wedge \mathbf{ord} \beta\}.
 \end{aligned}$$

Then

{array α (a, b)}

if a < b **then newvar** c **in**

({ $\exists x_1, \alpha_0, x_2. (a \mapsto x_1 * \text{array } \alpha_0(a + 1, b - 1) * b \mapsto x_2)$
 $\wedge x_1 \cdot \alpha_0 \cdot x_2 \sim \alpha$ }

newvar x1, x2, r **in**

(x1 := [a] ; x2 := [b] ;

if x1 > x2 **then** ([a] := x2 ; [b] := x1) **else skip** ;

r := (x1 + x2) \div 2 ;

{ $\exists x_1, \alpha_0, x_2. (a \mapsto x_1 * \text{array } \alpha_0(a + 1, b - 1) * b \mapsto x_2)$
 $\wedge x_1 \cdot \alpha_0 \cdot x_2 \sim \alpha \wedge x_1 \leq r \leq x_2$ }

partition(a + 1, b - 1, r; c)

{ $\exists x_1, \alpha_1, \alpha_2, x_2.$

(a \mapsto x1 * array $\alpha_1(a + 1, c)$ * array $\alpha_2(c + 1, b - 1)$ * b \mapsto x2)

$\wedge x_1 \cdot \alpha_1 \cdot \alpha_2 \cdot x_2 \sim \alpha \wedge x_1 \leq r \leq x_2 \wedge \{\alpha_1\} \leq^* r \wedge r <^* \{\alpha_2\}$) ;

{ $\exists \alpha_1, \alpha_2. (\text{array } \alpha_1(a, c) * \text{array } \alpha_2(c + 1, b))$

$\wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\}$ }

quicksort(a, c) ;

{ $\exists \beta_1, \alpha_2. (\text{array } \beta_1(a, c) * \text{array } \alpha_2(c + 1, b))$

$\wedge \beta_1 \cdot \alpha_2 \sim \alpha \wedge \{\beta_1\} \leq^* \{\alpha_2\} \wedge \mathbf{ord } \beta_1$ }

quicksort(c + 1, b)

{ $\exists \beta_1, \beta_2. (\text{array } \beta_1(a, c) * \text{array } \beta_2(c + 1, b))$

$\wedge \beta_1 \cdot \beta_2 \sim \alpha \wedge \{\beta_1\} \leq^* \{\beta_2\} \wedge \mathbf{ord } \beta_1 \wedge \mathbf{ord } \beta_2$ }

else skip

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

Quicksort (continued)

so that we may define

```
quicksort(a, b) =
  if a < b then newvar c in
    ( newvar x1, x2, r in
      ( x1 := [a] ; x2 := [b] ;
        if x1 > x2 then ([a] := x2 ; [b] := x1) else skip ;
        r := (x1 + x2) ÷ 2 ; partition(a + 1, b - 1, r; c) ) ;
      quicksort(a, c) ; quicksort(c + 1, b) )
    else skip.
```

Reasoning about the First Recursive Call

From the assumption

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

$$\text{quicksort}(a, b)$$

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

substitution (SUB) gives

$$\{ \text{array } \alpha_1 (a, c) \}$$

$$\text{quicksort}(a, c)$$

$$\{ \exists \beta_1. \text{array } \beta_1 (a, c) \wedge \beta_1 \sim \alpha_1 \wedge \mathbf{ord} \beta_1 \}.$$

Reasoning about the First Recursive Call (continued)

Then the frame rule (FR) gives the main step in

$$\begin{aligned}
 & \{(\text{array } \alpha_1 (\mathbf{a}, \mathbf{c}) * \text{array } \alpha_2 (\mathbf{c} + 1, \mathbf{b})) \\
 & \quad \wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\}\} \\
 & \{\text{array } \alpha_1 (\mathbf{a}, \mathbf{c}) \\
 & \quad * (\text{array } \alpha_2 (\mathbf{c} + 1, \mathbf{b}) \wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\})\} \\
 & \text{quicksort}(\mathbf{a}, \mathbf{c}) \\
 & \{(\exists \beta_1. \text{array } \beta_1 (\mathbf{a}, \mathbf{c}) \wedge \beta_1 \sim \alpha_1 \wedge \mathbf{ord} \beta_1) \\
 & \quad * (\text{array } \alpha_2 (\mathbf{c} + 1, \mathbf{b}) \wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\})\} \\
 & \{\exists \beta_1. (\text{array } \beta_1 (\mathbf{a}, \mathbf{c}) * \text{array } \alpha_2 (\mathbf{c} + 1, \mathbf{b})) \\
 & \quad \wedge \beta_1 \sim \alpha_1 \wedge \mathbf{ord} \beta_1 \wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\}\},
 \end{aligned}$$

and existential quantification (EQ) gives the main step in

$$\begin{aligned}
 & \{\exists \alpha_1, \alpha_2. (\text{array } \alpha_1 (\mathbf{a}, \mathbf{c}) * \text{array } \alpha_2 (\mathbf{c} + 1, \mathbf{b})) \\
 & \quad \wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\}\} \\
 & \text{quicksort}(\mathbf{a}, \mathbf{c}) ; \\
 & \{\exists \alpha_1, \alpha_2, \beta_1. (\text{array } \beta_1 (\mathbf{a}, \mathbf{c}) * \text{array } \alpha_2 (\mathbf{c} + 1, \mathbf{b})) \\
 & \quad \wedge \beta_1 \sim \alpha_1 \wedge \mathbf{ord} \beta_1 \wedge \alpha_1 \cdot \alpha_2 \sim \alpha \wedge \{\alpha_1\} \leq^* \{\alpha_2\}\} \\
 & \{\exists \beta_1, \alpha_2. (\text{array } \beta_1 (\mathbf{a}, \mathbf{c}) * \text{array } \alpha_2 (\mathbf{c} + 1, \mathbf{b})) \\
 & \quad \wedge \beta_1 \cdot \alpha_2 \sim \alpha \wedge \{\beta_1\} \leq^* \{\alpha_2\} \wedge \mathbf{ord} \beta_1\}.
 \end{aligned}$$

