CLASS NOTES FOR CS 819 C - SPRING 2002

4. Lists and Arrays

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

revised May 1, 2002

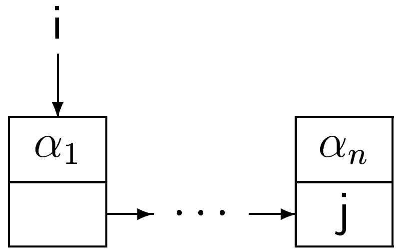
Sequences

When α and β are sequences, we write

- ϵ for the empty sequence.
- $[x]$ for the single-element sequence containing x. (We will omit the brackets when x is not a sequence.)
- $\alpha{\cdot}\beta$ for the composition of α followed by β.
- $\#\alpha$ for the length of α.
- α_i for the ith component of α.

An Inductive Definition: Singly-Linked Lists

$\mathsf{list}\ \alpha\,(\mathsf{i},\mathsf{j})$:



Definition

$$\mathsf{list}\ \epsilon\,(\mathsf{i},\mathsf{j}) \stackrel{\text{def}}{=} \mathbf{emp} \wedge \mathsf{i} = \mathsf{j}$$

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

Properties

$$\mathsf{list}\ \mathsf{a}\,(\mathsf{i},\mathsf{j}) \Leftrightarrow \mathsf{i} \mapsto \mathsf{a},\mathsf{j}$$

$$\mathsf{list}\ \alpha{\cdot}\beta\,(\mathsf{i},\mathsf{k}) \Leftrightarrow \exists \mathsf{j}.\ \mathsf{list}\ \alpha\,(\mathsf{i},\mathsf{j}) \ * \ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k})$$

$$\mathsf{list}\ \alpha{\cdot}\mathsf{b}\,(\mathsf{i},\mathsf{k}) \Leftrightarrow \exists \mathsf{j}.\ \mathsf{list}\ \alpha\,(\mathsf{i},\mathsf{j}) \ * \ \mathsf{j} \mapsto \mathsf{b},\mathsf{k}$$

Proof of the Composition Property

$$\mathsf{list}\ \alpha{\cdot}\beta\,(\mathsf{i},\mathsf{k}) \Leftrightarrow \exists \mathsf{j}.\ \mathsf{list}\ \alpha\,(\mathsf{i},\mathsf{j}) \ * \ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k})$$

The proof is by induction on the length of α.

When α is empty:

$$\begin{aligned}
&\exists \mathsf{j}.\ \mathsf{list}\ \epsilon\,(\mathsf{i},\mathsf{j}) \ * \ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k}) \\
&\quad \Leftrightarrow \exists \mathsf{j}.\ (\mathbf{emp} \wedge \mathsf{i} = \mathsf{j}) \ * \ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k}) \\
&\quad \Leftrightarrow \exists \mathsf{j}.\ (\mathbf{emp} \ * \ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k})) \wedge \mathsf{i} = \mathsf{j} \\
&\quad \Leftrightarrow \exists \mathsf{j}.\ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k}) \wedge \mathsf{i} = \mathsf{j} \\
&\quad \Leftrightarrow \mathsf{list}\ \beta\,(\mathsf{i},\mathsf{k}) \\
&\quad \Leftrightarrow \mathsf{list}\ \epsilon{\cdot}\beta\,(\mathsf{i},\mathsf{k})
\end{aligned}$$

When $\alpha = \mathsf{a}{\cdot}\alpha'$:

$$\begin{aligned}
&\exists \mathsf{j}.\ \mathsf{list}\ \mathsf{a}{\cdot}\alpha'\,(\mathsf{i},\mathsf{j}) \ * \ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k}) \\
&\quad \Leftrightarrow \exists \mathsf{j},\mathsf{l}.\ \mathsf{i} \mapsto \mathsf{a},\mathsf{l} \ * \ \mathsf{list}\ \alpha'\,(\mathsf{l},\mathsf{j}) \ * \ \mathsf{list}\ \beta\,(\mathsf{j},\mathsf{k}) \\
&\quad \Leftrightarrow \exists \mathsf{l}.\ \mathsf{i} \mapsto \mathsf{a},\mathsf{l} \ * \ \mathsf{list}\ \alpha'{\cdot}\beta\,(\mathsf{l},\mathsf{k}) \qquad \text{(induction hypothesis)} \\
&\quad \Leftrightarrow \mathsf{list}\ \mathsf{a}{\cdot}\alpha'{\cdot}\beta\,(\mathsf{i},\mathsf{k})
\end{aligned}$$

Cycles

When $\mathsf{list}\,\alpha_1 \cdots \alpha_n\,(\mathsf{i}_0, \mathsf{i}_n)$, we have

$$\exists \mathsf{i}_1, \ldots \mathsf{i}_{n-1}.\ \mathsf{i}_0 \mapsto \alpha_1, \mathsf{i}_1 \ * \ \mathsf{i}_1 \mapsto \alpha_2, \mathsf{i}_2 \ * \ \cdots \ * \ \mathsf{i}_{n-1} \mapsto \alpha_n, \mathsf{i}_n.$$

Then $\mathsf{i}_0, \ldots, \mathsf{i}_{n-1}$ are distinct locations, but i_n is not constrained. In particular, $\mathsf{list}\,\alpha_1 \cdots \alpha_n\,(\mathsf{i}, \mathsf{i})$ may hold for any $n \geq 0$.

Emptyness Conditions

$$\mathsf{list}\,\alpha\,(\mathsf{i}, \mathsf{j}) \Rightarrow (\mathsf{i} = \mathbf{nil} \Rightarrow (\alpha = \epsilon \wedge \mathsf{j} = \mathbf{nil}))$$

$$\mathsf{list}\,\alpha\,(\mathsf{i}, \mathsf{j}) \Rightarrow (\mathsf{i} \neq \mathsf{j} \Rightarrow \alpha \neq \epsilon).$$

In general, when $\mathsf{list}\,\alpha\,(\mathsf{i}, \mathsf{j})$ and $\mathsf{i} = \mathsf{j} \neq \mathbf{nil}$, α may be empty or nonempty. However,

$$\mathsf{list}\,\alpha\,(\mathsf{i}, \mathsf{j}) \ * \ \mathsf{j} \hookrightarrow - \Rightarrow (\mathsf{i} = \mathsf{j} \Leftrightarrow \alpha = \epsilon)$$

$$\mathsf{list}\,\alpha\,(\mathsf{i}, \mathsf{j}) \ * \ \mathsf{list}\,\beta\,(\mathsf{j}, \mathbf{nil}) \Rightarrow (\mathsf{i} = \mathsf{j} \Leftrightarrow \alpha = \epsilon).$$

Inserting a List Element

- At the beginning:

$$\{\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\}$$
$$\mathsf{k} := \textbf{cons}(\mathsf{a},\mathsf{i});$$
$$\{\mathsf{k} \mapsto \mathsf{a},\mathsf{i}\ *\ \textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\}$$
$$\{\exists \mathsf{i}.\ \mathsf{k} \mapsto \mathsf{a},\mathsf{i}\ *\ \textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\}$$
$$\{\textbf{list}\ \mathsf{a}{\cdot}\alpha\,(\mathsf{k},\mathsf{j})\}$$
$$\mathsf{i} := \mathsf{k}$$
$$\{\textbf{list}\ \mathsf{a}{\cdot}\alpha\,(\mathsf{i},\mathsf{j})\}$$

- At the end of a nonempty segment:

$$\{\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{k}\}$$
$$\mathsf{l} := \textbf{cons}(\mathsf{b},\mathsf{k})\ ;$$
$$\{\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}$$
$$\{\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a}\ *\ \mathsf{j}+1 \mapsto \mathsf{k}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}$$
$$\{\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a}\ *\ \mathsf{j}+1 \mapsto -\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}$$
$$[\mathsf{j}+1] := \mathsf{l}$$
$$\{\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a}\ *\ \mathsf{j}+1 \mapsto \mathsf{l}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}$$
$$\{\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{l}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}$$
$$\{\textbf{list}\ \alpha{\cdot}\mathsf{a}\,(\mathsf{i},\mathsf{l})\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}$$
$$\{\textbf{list}\ \alpha{\cdot}\mathsf{a}{\cdot}\mathsf{b}\,(\mathsf{i},\mathsf{k})\}$$

Deleting a List Element

- At the beginning:

$$
\begin{array}{l}
\{\mathbf{list}\ \mathsf{a}{\cdot}\alpha\,(\mathsf{i},\mathsf{k})\}\\
\{\exists \mathsf{j}.\ \mathsf{i} \mapsto \mathsf{a},\mathsf{j}\ *\ \mathbf{list}\ \alpha\,(\mathsf{j},\mathsf{k})\}\\
\{\mathsf{i} \mapsto \mathsf{a}\ *\ \exists \mathsf{j}.\ \mathsf{i}+1 \mapsto \mathsf{j}\ *\ \mathbf{list}\ \alpha\,(\mathsf{j},\mathsf{k})\}\\
\mathsf{j} := [\mathsf{i}+1]\ ;\\
\{\mathsf{i} \mapsto \mathsf{a}\ *\ \mathsf{i}+1 \mapsto \mathsf{j}\ *\ \mathbf{list}\ \alpha\,(\mathsf{j},\mathsf{k})\}\\
\mathbf{dispose}\ \mathsf{i}\ ;\\
\{\mathsf{i}+1 \mapsto \mathsf{j}\ *\ \mathbf{list}\ \alpha\,(\mathsf{j},\mathsf{k})\}\\
\mathbf{dispose}\ \mathsf{i}+1\ ;\\
\{\mathbf{list}\ \alpha\,(\mathsf{j},\mathsf{k})\}\\
\mathsf{i} := \mathsf{j}\\
\{\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{k})\}
\end{array}
$$

- At the end of a nonempty segment:

$$
\begin{array}{l}
\{\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathsf{k} \mapsto \mathsf{b},\mathsf{l}\}\\
[\mathsf{j}+1] := \mathsf{l}\ ;\\
\{\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{l}\ *\ \mathsf{k} \mapsto \mathsf{b},\mathsf{l}\}\\
\mathbf{dispose}\ \mathsf{k}\ ;\ \mathbf{dispose}\ \mathsf{k}+1\\
\{\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{l}\}\\
\{\mathbf{list}\ \alpha{\cdot}\mathsf{a}\,(\mathsf{i},\mathsf{l})\}
\end{array}
$$

Reversing a List

$$\{\mathbf{list}\,\gamma\,(\mathsf{i},\mathbf{nil})\}$$
$$\{\mathbf{list}\,\gamma\,(\mathsf{i},\mathbf{nil})\ *\ (\mathbf{emp} \wedge \mathbf{nil} = \mathbf{nil})\}$$
$$\mathsf{j} := \mathbf{nil}\,;$$
$$\{\mathbf{list}\,\gamma\,(\mathsf{i},\mathbf{nil})\ *\ (\mathbf{emp} \wedge \mathsf{j} = \mathbf{nil})\}$$
$$\{\mathbf{list}\,\gamma\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\,\epsilon\,(\mathsf{j},\mathbf{nil})\}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\,\alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\,\beta\,(\mathsf{j},\mathbf{nil})) \wedge \gamma^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathbf{while}\ \mathsf{i} \neq \mathbf{nil}\ \mathbf{do}$$
$$\big(\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\,\mathsf{a}{\cdot}\alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\,\beta\,(\mathsf{j},\mathbf{nil})) \wedge \gamma^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta,\mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\,\beta\,(\mathsf{j},\mathbf{nil}))$$
$$\wedge\,\gamma^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i}+1]\,;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\,\beta\,(\mathsf{j},\mathbf{nil}))$$
$$\wedge\,\gamma^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i}+1] := \mathsf{j}\,;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{j}\ *\ \mathbf{list}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\,\beta\,(\mathsf{j},\mathbf{nil}))$$
$$\wedge\,\gamma^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\,\mathsf{a}{\cdot}\beta\,(\mathsf{i},\mathbf{nil})) \wedge \gamma^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\,\beta\,(\mathsf{i},\mathbf{nil})) \wedge \gamma^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\,;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\,\alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\,\beta\,(\mathsf{j},\mathbf{nil})) \wedge \gamma^\dagger = \alpha^\dagger{\cdot}\beta\}\big)$$
$$\{\exists\alpha,\beta.\ \mathbf{list}\,\beta\,(\mathsf{j},\mathbf{nil}) \wedge \gamma^\dagger = \alpha^\dagger{\cdot}\beta \wedge \alpha = \epsilon\}$$
$$\{\mathbf{list}\,\gamma^\dagger\,(\mathsf{j},\mathbf{nil})\}$$

A Cyclic Buffer

$$\exists \beta.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathsf{i})) \wedge \mathsf{m} = \#\alpha \wedge \mathsf{n} = \#\alpha + \#\beta$$

(If the buffer is empty ($\#\alpha = 0$) or full ($\#\beta = 0$), then $\mathsf{i} = \mathsf{j}$.)

To insert an element:

$$\{\exists \beta.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathsf{i})) \wedge \mathsf{m} = \#\alpha \wedge \mathsf{n} = \#\alpha + \#\beta \wedge$$
$$\mathsf{n} - \mathsf{m} > 0\}$$

$$\{\exists \mathsf{b}, \beta.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathbf{list}\ \mathsf{b}{\cdot}\beta\,(\mathsf{j},\mathsf{i})) \wedge$$
$$\mathsf{m} = \#\alpha \wedge \mathsf{n} = \#\alpha + \#\mathsf{b}{\cdot}\beta\}$$

$$\{\exists \beta, \mathsf{j}''.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto -, \mathsf{j}''\ *\ \mathbf{list}\ \beta\,(\mathsf{j}'',\mathsf{i})) \wedge$$
$$\mathsf{m} = \#\alpha \wedge \mathsf{n} - 1 = \#\alpha + \#\beta\}$$

$$[\mathsf{j}] := \mathsf{x}\,;$$

$$\{\exists \beta, \mathsf{j}''.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{x}, \mathsf{j}''\ *\ \mathbf{list}\ \beta\,(\mathsf{j}'',\mathsf{i})) \wedge$$
$$\mathsf{m} = \#\alpha \wedge \mathsf{n} - 1 = \#\alpha + \#\beta\}$$

$$\{\exists \beta, \mathsf{j}''.\ \mathsf{j} + 1 \mapsto \mathsf{j}''\ *\ ((\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{x}\ *\ \mathbf{list}\ \beta\,(\mathsf{j}'',\mathsf{i})) \wedge$$
$$\mathsf{m} = \#\alpha \wedge \mathsf{n} - 1 = \#\alpha + \#\beta)\}$$

$$\mathsf{j} := [\mathsf{j} + 1]$$

$$\{\exists \beta, \mathsf{j}'.\ \mathsf{j}' + 1 \mapsto \mathsf{j}\ *\ ((\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j}')\ *\ \mathsf{j}' \mapsto \mathsf{x}\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathsf{i})) \wedge$$
$$\mathsf{m} = \#\alpha \wedge \mathsf{n} - 1 = \#\alpha + \#\beta)\}$$

$$\{\exists \beta, \mathsf{j}'.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j}')\ *\ \mathsf{j}' \mapsto \mathsf{x}, \mathsf{j}\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathsf{i})) \wedge$$
$$\mathsf{m} = \#\alpha \wedge \mathsf{n} - 1 = \#\alpha + \#\beta\}$$

$$\{\exists \beta.\ (\mathbf{list}\ \alpha{\cdot}\mathsf{x}\,(\mathsf{i},\mathsf{j})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathsf{i})) \wedge \mathsf{m} + 1 = \#\alpha{\cdot}\mathsf{x} \wedge \mathsf{n} = \#\alpha{\cdot}\mathsf{x} + \#\beta\}$$

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

$$\{\exists \beta.\ (\mathbf{list}\ \alpha{\cdot}\mathsf{x}\,(\mathsf{i},\mathsf{j})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathsf{i})) \wedge \mathsf{m} = \#\alpha{\cdot}\mathsf{x} \wedge \mathsf{n} = \#\alpha{\cdot}\mathsf{x} + \#\beta\}$$

Note the use of (LKLG) with r taken as $((\mathbf{list}\ \alpha\,(\mathsf{i},\mathsf{j}')\ *\ \mathsf{j}' \mapsto \mathsf{x}\ *\ \mathbf{list}\ \beta\,(\mathsf{j}'',\mathsf{i})) \wedge \mathsf{m} = \#\alpha \wedge \mathsf{n} - 1 = \#\alpha + \#\beta)$.

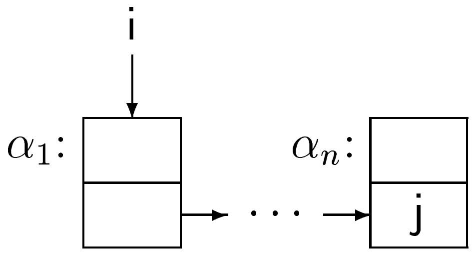
A Cyclic Buffer (continued)

To remove an element:

$$\{\exists\beta.\ (\textbf{list}\ \mathsf{a}{\cdot}\alpha\,(\mathsf{i},\mathsf{j})\ *\ \textbf{list}\ \beta\,(\mathsf{j},\mathsf{i}))\ \wedge$$
$$\quad \mathsf{m} = \#\mathsf{a}\cdot\alpha \wedge \mathsf{n} = \#\mathsf{a}{\cdot}\alpha + \#\beta \wedge \mathsf{m} > 0\}$$
$$\{\exists\beta,\mathsf{i}''.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{i}''\ *\ \textbf{list}\ \alpha\,(\mathsf{i}'',\mathsf{j})\ *\ \textbf{list}\ \beta\,(\mathsf{j},\mathsf{i}))\ \wedge$$
$$\quad \mathsf{m}-1 = \#\alpha \wedge \mathsf{n}-1 = \#\alpha + \#\beta\}$$
$$\{\exists\beta,\mathsf{i}'',\mathsf{y}.\ (\mathsf{i} \mapsto \mathsf{y},\mathsf{i}''\ *\ \textbf{list}\ \alpha\,(\mathsf{i}'',\mathsf{j})\ *\ \textbf{list}\ \beta\,(\mathsf{j},\mathsf{i}))\ \wedge$$
$$\quad \mathsf{m}-1 = \#\alpha \wedge \mathsf{n}-1 = \#\alpha + \#\beta \wedge \mathsf{y} = \mathsf{a}\}$$
$$\mathsf{y} := [\mathsf{i}]$$
$$\{\exists\beta,\mathsf{i}''.\ (\mathsf{i} \mapsto \mathsf{y},\mathsf{i}''\ *\ \textbf{list}\ \alpha\,(\mathsf{i}'',\mathsf{j})\ *\ \textbf{list}\ \beta\,(\mathsf{j},\mathsf{i}))\ \wedge$$
$$\quad \mathsf{m}-1 = \#\alpha \wedge \mathsf{n}-1 = \#\alpha + \#\beta \wedge \mathsf{y} = \mathsf{a}\}$$
$$\mathsf{i} := [\mathsf{i}+1]$$
$$\{\exists\beta,\mathsf{i}'.\ (\mathsf{i}' \mapsto \mathsf{y},\mathsf{i}\ *\ \textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \textbf{list}\ \beta\,(\mathsf{j},\mathsf{i}'))\ \wedge$$
$$\quad \mathsf{m}-1 = \#\alpha \wedge \mathsf{n}-1 = \#\alpha + \#\beta \wedge \mathsf{y} = \mathsf{a}\}$$
$$\{\exists\beta.\ (\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \textbf{list}\ \beta{\cdot}\mathsf{y}\,(\mathsf{j},\mathsf{i}))\ \wedge$$
$$\quad \mathsf{m}-1 = \#\alpha \wedge \mathsf{n} = \#\alpha + \#\beta{\cdot}\mathsf{y} \wedge \mathsf{y} = \mathsf{a}\}$$
$$\{\exists\beta.\ (\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \textbf{list}\ \beta\,(\mathsf{j},\mathsf{i}))\ \wedge$$
$$\quad \mathsf{m}-1 = \#\alpha \wedge \mathsf{n} = \#\alpha + \#\beta \wedge \mathsf{y} = \mathsf{a}\}$$
$$\mathsf{m} := \mathsf{m}-1$$
$$\{\exists\beta.\ (\textbf{list}\ \alpha\,(\mathsf{i},\mathsf{j})\ *\ \textbf{list}\ \beta\,(\mathsf{j},\mathsf{i})) \wedge \mathsf{m} = \#\alpha \wedge \mathsf{n} = \#\alpha + \#\beta \wedge \mathsf{y} = \mathsf{a}\}$$

Bornat's Lists

$\mathsf{listN}\,\alpha\,(\mathsf{i},\mathsf{j})$:



Definition

$$\mathsf{listN}\,\epsilon\,(\mathsf{i},\mathsf{j}) \stackrel{\text{def}}{=} \mathbf{emp} \wedge \mathsf{i} = \mathsf{j}$$

$$\mathsf{listN}\,\mathsf{a}{\cdot}\alpha\,(\mathsf{i},\mathsf{k}) \stackrel{\text{def}}{=} \mathsf{a} = \mathsf{i} \wedge \exists \mathsf{j}.\ \mathsf{i} + 1 \mapsto \mathsf{j}\ *\ \mathsf{listN}\,\alpha\,(\mathsf{j},\mathsf{k})$$

Properties

$$\mathsf{listN}\,\mathsf{a}\,(\mathsf{i},\mathsf{j}) \Leftrightarrow \mathsf{a} = \mathsf{i} \wedge \mathsf{i} + 1 \mapsto \mathsf{j}$$

$$\mathsf{listN}\,\alpha{\cdot}\beta\,(\mathsf{i},\mathsf{k}) \Leftrightarrow \exists \mathsf{j}.\ \mathsf{listN}\,\alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{listN}\,\beta\,(\mathsf{j},\mathsf{k})$$

$$\mathsf{listN}\,\alpha{\cdot}\mathsf{b}\,(\mathsf{i},\mathsf{k}) \Leftrightarrow \exists \mathsf{j}.\ \mathsf{listN}\,\alpha\,(\mathsf{i},\mathsf{j})\ *\ (\mathsf{b} = \mathsf{k} \wedge \mathsf{j} + 1 \mapsto \mathsf{k})$$

Emptyness Conditions

$$\mathsf{listN}\,\alpha\,(\mathsf{i},\mathsf{j}) \Rightarrow (\mathsf{i} = \mathbf{nil} \Rightarrow (\alpha = \epsilon \wedge \mathsf{j} = \mathbf{nil}))$$

$$\mathsf{listN}\,\alpha\,(\mathsf{i},\mathsf{j}) \Rightarrow (\mathsf{i} \neq \mathsf{j} \Rightarrow \alpha \neq \epsilon)$$

$$\mathsf{listN}\,\alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{j} + 1 \hookrightarrow - \Rightarrow (\mathsf{i} = \mathsf{j} \Leftrightarrow \alpha = \epsilon)$$

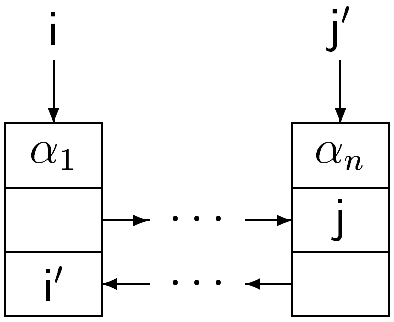
$$\mathsf{listN}\,\alpha\,(\mathsf{i},\mathsf{j})\ *\ \mathsf{listN}\,\beta\,(\mathsf{j},\mathbf{nil}) \Rightarrow (\mathsf{i} = \mathsf{j} \Leftrightarrow \alpha = \epsilon).$$

Reversing a Bornat List

$$
\begin{array}{l}
\{\mathbf{listN}\,\gamma\,(\mathsf{i},\mathbf{nil})\}\\
\{\mathbf{listN}\,\gamma\,(\mathsf{i},\mathbf{nil})\ *\ (\mathbf{emp}\wedge\mathbf{nil}=\mathbf{nil})\}\\
\mathsf{j} := \mathbf{nil}\,;\\
\{\mathbf{listN}\,\gamma\,(\mathsf{i},\mathbf{nil})\ *\ (\mathbf{emp}\wedge\mathsf{j}=\mathbf{nil})\}\\
\{\mathbf{listN}\,\gamma\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{listN}\,\epsilon\,(\mathsf{j},\mathbf{nil})\}\\
\{\exists\alpha,\beta.\ (\mathbf{listN}\,\alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{listN}\,\beta\,(\mathsf{j},\mathbf{nil}))\wedge\gamma^\dagger=\alpha^\dagger{\cdot}\beta\}\\
\mathbf{while}\ \mathsf{i}\neq\mathbf{nil}\ \mathbf{do}\\
\quad\big(\{\exists\alpha,\beta.\ (\mathbf{listN}\,\mathsf{i}{\cdot}\alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{listN}\,\beta\,(\mathsf{j},\mathbf{nil}))\wedge\gamma^\dagger=(\mathsf{i}{\cdot}\alpha)^\dagger{\cdot}\beta\}\\
\quad\ \{\exists\alpha,\beta,\mathsf{k}.\ (\mathsf{i}+1\mapsto\mathsf{k}\ *\ \mathbf{listN}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{listN}\,\beta\,(\mathsf{j},\mathbf{nil}))\\
\qquad\quad\wedge\gamma^\dagger=(\mathsf{i}{\cdot}\alpha)^\dagger{\cdot}\beta\}\\
\quad\ \mathsf{k} := [\mathsf{i}+1]\,;\\
\quad\ \{\exists\alpha,\beta.\ (\mathsf{i}+1\mapsto\mathsf{k}\ *\ \mathbf{listN}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{listN}\,\beta\,(\mathsf{j},\mathbf{nil}))\\
\qquad\quad\wedge\gamma^\dagger=(\mathsf{i}{\cdot}\alpha)^\dagger{\cdot}\beta\}\\
\quad\ [\mathsf{i}+1] := \mathsf{j}\,;\\
\quad\ \{\exists\alpha,\beta.\ (\mathsf{i}+1\mapsto\mathsf{j}\ *\ \mathbf{listN}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{listN}\,\beta\,(\mathsf{j},\mathbf{nil}))\\
\qquad\quad\wedge\gamma^\dagger=(\mathsf{i}{\cdot}\alpha)^\dagger{\cdot}\beta\}\\
\quad\ \{\exists\alpha,\beta.\ (\mathbf{listN}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{listN}\,\mathsf{i}{\cdot}\beta\,(\mathsf{i},\mathbf{nil}))\wedge\gamma^\dagger=\alpha^\dagger{\cdot}\mathsf{i}{\cdot}\beta\}\\
\quad\ \{\exists\alpha,\beta.\ (\mathbf{listN}\,\alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{listN}\,\beta\,(\mathsf{i},\mathbf{nil}))\wedge\gamma^\dagger=\alpha^\dagger{\cdot}\beta\}\\
\quad\ \mathsf{j} := \mathsf{i}\,;\ \mathsf{i} := \mathsf{k}\\
\quad\ \{\exists\alpha,\beta.\ (\mathbf{listN}\,\alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{listN}\,\beta\,(\mathsf{j},\mathbf{nil}))\wedge\gamma^\dagger=\alpha^\dagger{\cdot}\beta\}\big)\\
\{\exists\alpha,\beta.\ \mathbf{listN}\,\beta\,(\mathsf{j},\mathbf{nil})\wedge\gamma^\dagger=\alpha^\dagger{\cdot}\beta\wedge\alpha=\epsilon\}\\
\{\mathbf{listN}\,\gamma^\dagger\,(\mathsf{j},\mathbf{nil})\}
\end{array}
$$

An Inductive Definition: Doubly-Linked Lists

$\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}')$:



Definition

$$\mathsf{dlist}\,\epsilon\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}') \stackrel{\text{def}}{=} \mathbf{emp} \land \mathsf{i} = \mathsf{j} \land \mathsf{i}' = \mathsf{j}'$$

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

Properties

$$\mathsf{dlist}\,\mathsf{a}\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}') \Leftrightarrow \mathsf{i} \mapsto \mathsf{a},\mathsf{j},\mathsf{i}' \land \mathsf{i} = \mathsf{j}'$$

$$\mathsf{dlist}\,\alpha{\cdot}\beta\,(\mathsf{i},\mathsf{i}',\mathsf{k},\mathsf{k}') \Leftrightarrow \exists \mathsf{j},\mathsf{j}'.\ \mathsf{dlist}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}') \ * \ \mathsf{dlist}\,\beta\,(\mathsf{j},\mathsf{j}',\mathsf{k},\mathsf{k}')$$

$$\mathsf{dlist}\,\alpha{\cdot}\mathsf{b}\,(\mathsf{i},\mathsf{i}',\mathsf{k},\mathsf{k}') \Leftrightarrow \exists \mathsf{j}'.\ \mathsf{dlist}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathsf{k}',\mathsf{j}') \ * \ \mathsf{k}' \mapsto \mathsf{b},\mathsf{k},\mathsf{j}'.$$

Emptyness Conditions

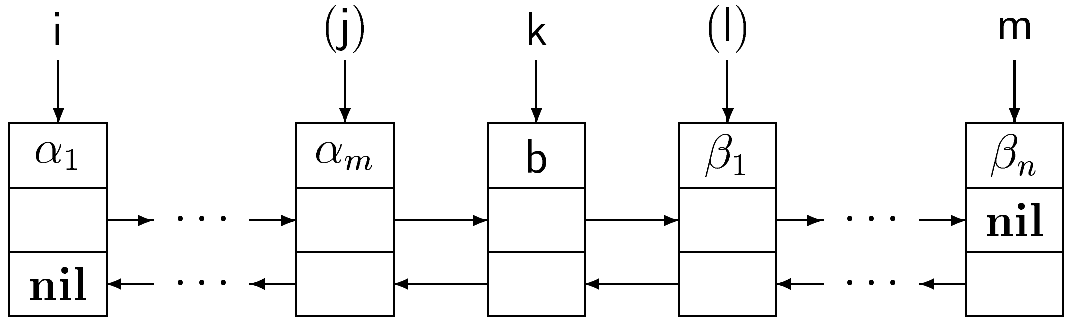
$$\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}') \ \land \quad \mathsf{i} = \mathbf{nil} \ \Rightarrow \ (\alpha = \epsilon \land \mathsf{j} = \mathbf{nil} \land \mathsf{i}' = \mathsf{j}')$$

$$\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}') \ \land \quad \mathsf{j}' = \mathbf{nil} \ \Rightarrow \ (\alpha = \epsilon \land \mathsf{i}' = \mathbf{nil} \land \mathsf{i} = \mathsf{j})$$

$$\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}') \ \land \quad \mathsf{i} \neq \mathsf{j} \ \Rightarrow \ \alpha \neq \epsilon$$

$$\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathsf{j},\mathsf{j}') \ \land \quad \mathsf{i}' \neq \mathsf{j}' \ \Rightarrow \ \alpha \neq \epsilon.$$

Deleting an Element from a Doubly-Linked List



$\{\exists \mathsf{j}, \mathsf{l}.\ \mathsf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j}) \ * \ \mathsf{k} \mapsto \mathsf{b}, \mathsf{l}, \mathsf{j} \ * \ \mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{l} := [\mathsf{k}+1]\ ;\ \mathsf{j} := [\mathsf{k}+2]\ ;$

$\{\mathsf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j}) \ * \ \mathsf{k} \mapsto \mathsf{b}, \mathsf{l}, \mathsf{j} \ * \ \mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\mathbf{dispose}\ \mathsf{k}\ ;\ \mathbf{dispose}\ \mathsf{k}+1\ ;\ \mathbf{dispose}\ \mathsf{k}+2\ ;$

$\{\mathsf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j}) \ * \ \mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\mathbf{if}\ \mathsf{j} = \mathbf{nil}\ \mathbf{then}$

$\quad\{\mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m}) \wedge \mathsf{i} = \mathsf{k} \wedge \mathbf{nil} = \mathsf{j} \wedge \alpha = \epsilon\}$

$\quad\mathsf{i} := \mathsf{l}$

$\quad\{\mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m}) \wedge \mathsf{i} = \mathsf{l} \wedge \mathbf{nil} = \mathsf{j} \wedge \alpha = \epsilon\}$

$\mathbf{else}$

$\quad\{\exists \alpha', \mathsf{a}, \mathsf{n}.\ (\mathsf{dlist}\ \alpha'\ (\mathsf{i}, \mathbf{nil}, \mathsf{j}, \mathsf{n}) \ * \ \mathsf{j} \mapsto \mathsf{a}, \mathsf{k}, \mathsf{n}$

$\qquad * \ \mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})) \wedge \alpha = \alpha'{\cdot}\mathsf{a}\}$

$\quad[\mathsf{j}+1] := \mathsf{l}\ ;$

$\quad\{\exists \alpha', \mathsf{a}, \mathsf{n}.\ (\mathsf{dlist}\ \alpha'\ (\mathsf{i}, \mathbf{nil}, \mathsf{j}, \mathsf{n}) \ * \ \mathsf{j} \mapsto \mathsf{a}, \mathsf{l}, \mathsf{n}$

$\qquad * \ \mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})) \wedge \alpha = \alpha'{\cdot}\mathsf{a}\}$

$\{\mathsf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j}) \ * \ \mathsf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\qquad\vdots$

Deleting from a Doubly-Linked List (continued)

$$\vdots$$

$\{\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \ * \ \mathsf{dlist}\,\beta\,(\mathsf{l},\mathsf{k},\mathbf{nil},\mathsf{m})\}$

if $\mathsf{l} = \mathbf{nil}$ **then**

$\quad\{\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \wedge \mathsf{l} = \mathbf{nil} \wedge \mathsf{k} = \mathsf{m} \wedge \beta = \epsilon\}$

$\quad\mathsf{m} := \mathsf{j}$

$\quad\{\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \wedge \mathsf{l} = \mathbf{nil} \wedge \mathsf{j} = \mathsf{m} \wedge \beta = \epsilon\}$

else

$\quad\{\exists \mathsf{a}, \beta', \mathsf{n}.\ (\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \ * \ \mathsf{l} \mapsto \mathsf{a},\mathsf{n},\mathsf{k}$

$\qquad * \ \mathsf{dlist}\,\beta'\,(\mathsf{n},\mathsf{l},\mathbf{nil},\mathsf{m})) \wedge \beta = \mathsf{a}\cdot\beta'\}$

$\quad[\mathsf{l} + 2] := \mathsf{j}$

$\quad\{\exists \mathsf{a}, \beta', \mathsf{n}.\ (\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \ * \ \mathsf{l} \mapsto \mathsf{a},\mathsf{n},\mathsf{j}$

$\qquad * \ \mathsf{dlist}\,\beta'\,(\mathsf{n},\mathsf{l},\mathbf{nil},\mathsf{m})) \wedge \beta = \mathsf{a}\cdot\beta'\}$

$\{\mathsf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \ * \ \mathsf{dlist}\,\beta\,(\mathsf{l},\mathsf{j},\mathbf{nil},\mathsf{m})\}$

$\{\mathsf{dlist}\,\alpha\cdot\beta\,(\mathsf{i},\mathbf{nil},\mathbf{nil},\mathsf{m})\}$

Inference for Simple Procedures

A simple procedure definition has the form

$$h(x_1, \cdots, x_m; y_1, \cdots, y_n) = c,$$

where $y_1, \cdots, y_n$ are the free variables assigned by c, and $x_1, \cdots, x_m$ are the other free variables of c.

If one can prove the specification $\{p\}\ c\ \{q\}$, then one can assume the specification $\{p\}\ h(x_1, \cdots, x_m; y_1, \cdots, y_n)\ \{q\}$ in proving a specification of a command that contains calls of h:

Simple Procedures (SPROC)

When $h(x_1, \cdots, x_m; y_1, \cdots, y_n) = c$,

$$\frac{\{p\}\ c\ \{q\}}{\{p\}\ h(x_1, \cdots, x_m; y_1, \cdots, y_n)\ \{q\}.}$$

From the conclusion of this rule, one can reason about other calls by using the rule for free variable substitution (FVS), and

$$\mathrm{FA}(h(x_1, \cdots, x_m; y_1, \cdots, y_n)) = y_1, \cdots, y_n.$$

For partial correctness, if c contains recursive calls, one can assume $\{p\}\ h(x_1, \cdots, x_m; y_1, \cdots, y_n)\ \{q\}$ in proving $\{p\}\ c\ \{q\}$:

Simple Recursive Procedures (SRPROC)

When $h(x_1, \cdots, x_m; y_1, \cdots, y_n) = c$,

$$\frac{\begin{array}{c}\{p\}\ h(x_1, \cdots, x_m; y_1, \cdots, y_n)\ \{q\} \\ \vdots \\ \{p\}\ c\ \{q\}\end{array}}{\{p\}\ h(x_1, \cdots, x_m; y_1, \cdots, y_n)\ \{q\}.}$$

Since

$$
\begin{array}{l}
\{\mathbf{dlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{j}_0, \mathsf{j}')\} \\
\mathbf{if}\ \mathsf{j}' = \mathbf{nil}\ \mathbf{then} \\
\quad \{\mathbf{dlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{j}_0, \mathsf{j}') \wedge \mathsf{i} = \mathsf{j}_0\} \\
\quad \mathsf{j} := \mathsf{i} \\
\mathbf{else} \\
\quad \{\exists \alpha', \mathsf{b}, \mathsf{k}'.\ \ \alpha = \alpha'{\cdot}\mathsf{b} \wedge (\mathbf{dlist}\ \alpha'\,(\mathsf{i}, \mathbf{nil}, \mathsf{j}', \mathsf{k}')\ *\ \mathsf{j}' \mapsto \mathsf{b}, \mathsf{j}_0, \mathsf{k}')\} \\
\quad \mathsf{j} := [\mathsf{j}' + 1] \\
\quad \{\exists \alpha', \mathsf{b}, \mathsf{k}'.\ \ \alpha = \alpha'{\cdot}\mathsf{b}\ \wedge \\
\qquad (\mathbf{dlist}\ \alpha'\,(\mathsf{i}, \mathbf{nil}, \mathsf{j}', \mathsf{k}')\ *\ \mathsf{j}' \mapsto \mathsf{b}, \mathsf{j}_0, \mathsf{k}') \wedge \mathsf{j} = \mathsf{j}_0\} \\
\{\mathbf{dlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{j}_0, \mathsf{j}') \wedge \mathsf{j} = \mathsf{j}_0\},
\end{array}
$$

the procedure

$$
\begin{array}{l}
\mathsf{lookuprpt}(\mathsf{i}, \mathsf{j}'; \mathsf{j}) = \\
\quad \mathbf{if}\ \mathsf{j}' = \mathbf{nil}\ \mathbf{then}\ \mathsf{j} := \mathsf{i}\ \mathbf{else}\ \mathsf{j} := [\mathsf{j}' + 1]
\end{array}
$$

satisfies

$$
\begin{array}{l}
\{\mathbf{dlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{j}_0, \mathsf{j}')\} \\
\mathsf{lookuprpt}(\mathsf{i}, \mathsf{j}'; \mathsf{j}) \\
\{\mathbf{dlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{j}_0, \mathsf{j}') \wedge \mathsf{j} = \mathsf{j}_0\}.
\end{array}
$$

A Set-pointer Procedure for Doubly-linked Lists

Since

$$
\begin{array}{l}
\{\exists \mathsf{j}.\ \ \mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j},\mathsf{j}')\}\\
\mathbf{if}\ \mathsf{j}' = \mathbf{nil}\ \mathbf{then}\\
\quad \{\alpha = \epsilon \wedge \mathbf{emp} \wedge \mathsf{j}' = \mathbf{nil}\}\\
\quad \mathsf{i} := \mathsf{j}\\
\quad \{\alpha = \epsilon \wedge \mathbf{emp} \wedge \mathsf{j}' = \mathbf{nil} \wedge \mathsf{i} = \mathsf{j}\}\\
\mathbf{else}\\
\quad \{\exists \alpha', \mathsf{b}, \mathsf{j}, \mathsf{k}'.\ \ \alpha = \alpha'{\cdot}\mathsf{b} \wedge (\mathbf{dlist}\,\alpha'\,(\mathsf{i},\mathbf{nil},\mathsf{j}',\mathsf{k}')\ *\ \mathsf{j}' \mapsto \mathsf{b},\mathsf{j},\mathsf{k}')\}\\
\quad [\mathsf{j}'+1] := \mathsf{j}\\
\quad \{\exists \alpha', \mathsf{b}, \mathsf{k}'.\ \ \alpha = \alpha'{\cdot}\mathsf{b} \wedge (\mathbf{dlist}\,\alpha'\,(\mathsf{i},\mathbf{nil},\mathsf{j}',\mathsf{k}')\ *\ \mathsf{j}' \mapsto \mathsf{b},\mathsf{j},\mathsf{k}')\}\\
\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j},\mathsf{j}')\},
\end{array}
$$

the procedure

$$
\begin{array}{l}
\mathsf{setrpt}(\mathsf{j},\mathsf{j}';\mathsf{i}) =\\
\quad \mathbf{if}\ \mathsf{j}' = \mathbf{nil}\ \mathbf{then}\ \mathsf{i} := \mathsf{j}\ \mathbf{else}\ [\mathsf{j}'+1] := \mathsf{j}
\end{array}
$$

satisfies

$$
\begin{array}{l}
\{\exists \mathsf{j}.\ \ \mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j},\mathsf{j}')\}\\
\mathsf{setrpt}(\mathsf{j},\mathsf{j}';\mathsf{i})\\
\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j},\mathsf{j}')\}.
\end{array}
$$

Upon Reflection

The procedure

$$\begin{array}{l}\mathsf{lookuplpt}(\mathsf{i},\mathsf{j}';\mathsf{i}') = \\ \qquad \textbf{if } \mathsf{i} = \textbf{nil then } \mathsf{i}' := \mathsf{j}' \textbf{ else } \mathsf{i}' := [\mathsf{i}+2]\end{array}$$

satisfies

$$\begin{array}{l}\{\textbf{dlist}\ \alpha\,(\mathsf{i},\mathsf{i}'_0,\textbf{nil},\mathsf{j}')\} \\ \mathsf{lookuplpt}(\mathsf{i},\mathsf{j}';\mathsf{i}') \\ \{\textbf{dlist}\ \alpha\,(\mathsf{i},\mathsf{i}'_0,\textbf{nil},\mathsf{j}') \wedge \mathsf{i}' = \mathsf{i}'_0\},\end{array}$$

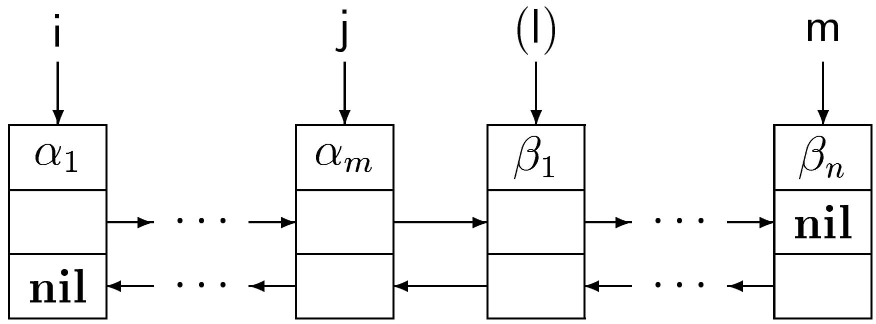
and the procedure

$$\begin{array}{l}\mathsf{setlpt}(\mathsf{i},\mathsf{i}';\mathsf{j}') = \\ \qquad \textbf{if } \mathsf{i} = \textbf{nil then } \mathsf{j}' := \mathsf{i}' \textbf{ else } [\mathsf{i}+2] := \mathsf{i}'\end{array}$$

satisfies

$$\begin{array}{l}\{\exists \mathsf{i}'.\ \ \textbf{dlist}\ \alpha\,(\mathsf{i},\mathsf{i}',\textbf{nil},\mathsf{j}')\} \\ \mathsf{setlpt}(\mathsf{i},\mathsf{i}';\mathsf{j}') \\ \{\textbf{dlist}\ \alpha\,(\mathsf{i},\mathsf{i}',\textbf{nil},\mathsf{j}')\}.\end{array}$$

Inserting an Element into a Doubly-Linked List



$\{\exists \mathsf{l}.\ \ \mathbf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j})\ *\ \mathbf{dlist}\ \beta\ (\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{lookuprpt}(\mathsf{i}, \mathsf{j}; \mathsf{l})$;

$\{\mathbf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j})\ *\ \mathbf{dlist}\ \beta\ (\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{k} := \mathbf{cons}(\mathsf{a}, \mathsf{l}, \mathsf{j})$;

$\{\mathbf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j})\ *\ \mathsf{k} \mapsto \mathsf{a}, \mathsf{l}, \mathsf{j}\ *\ \mathbf{dlist}\ \beta\ (\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{setrpt}(\mathsf{k}, \mathsf{j}; \mathsf{i})$;

$\{\mathbf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j})\ *\ \mathsf{k} \mapsto \mathsf{a}, \mathsf{l}, \mathsf{j}\ *\ \mathbf{dlist}\ \beta\ (\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{setlpt}(\mathsf{l}, \mathsf{k}; \mathsf{m})$

$\{\mathbf{dlist}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j})\ *\ \mathsf{k} \mapsto \mathsf{a}, \mathsf{l}, \mathsf{j}\ *\ \mathbf{dlist}\ \beta\ (\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\{\mathbf{dlist}\ \alpha{\cdot}\mathsf{a}{\cdot}\beta\ (\mathsf{i}, \mathbf{nil}, \mathbf{nil}, \mathsf{m})\}$

Deriving the Specification of the Call of lookuprpt

From:

$$\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j}_0,\mathsf{j}')\}$$
$$\mathsf{lookuprpt}(\mathsf{i},\mathsf{j}';\mathsf{j})$$
$$\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j}_0,\mathsf{j}') \wedge \mathsf{j} = \mathsf{j}_0\},$$

using Free Variable Substitution (FVS):

$$\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j}_0,\mathsf{j})\}$$
$$\mathsf{lookuprpt}(\mathsf{i},\mathsf{j};\mathsf{l})$$
$$\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j}_0,\mathsf{j}) \wedge \mathsf{l} = \mathsf{j}_0\},$$

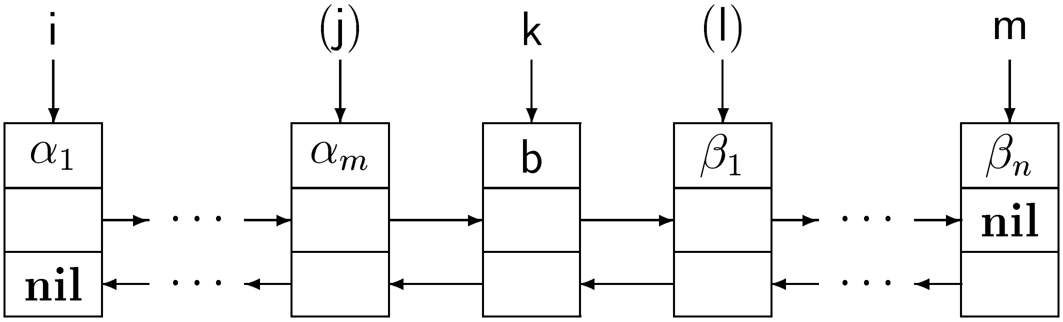
using the Frame Rule (FR) and an obvious property of equality:

$$\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j}_0,\mathsf{j}) \;*\; \mathbf{dlist}\,\beta\,(\mathsf{j}_0,\mathsf{j},\mathbf{nil},\mathsf{m})\}$$
$$\mathsf{lookuprpt}(\mathsf{i},\mathsf{j};\mathsf{l})$$
$$\{(\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j}_0,\mathsf{j}) \;*\; \mathbf{dlist}\,\beta\,(\mathsf{j}_0,\mathsf{j},\mathbf{nil},\mathsf{m})) \wedge \mathsf{l} = \mathsf{j}_0\}$$
$$\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \;*\; \mathbf{dlist}\,\beta\,(\mathsf{l},\mathsf{j},\mathbf{nil},\mathsf{m})\},$$

using the rules for Existential Quantification (EQ), Renaming (RN), and the elimination of a vacuous quantifier:

$$\{\exists \mathsf{l}.\; \mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \;*\; \mathbf{dlist}\,\beta\,(\mathsf{l},\mathsf{j},\mathbf{nil},\mathsf{m})\}$$
$$\{\exists \mathsf{j}_0.\; \mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j}_0,\mathsf{j}) \;*\; \mathbf{dlist}\,\beta\,(\mathsf{j}_0,\mathsf{j},\mathbf{nil},\mathsf{m})\}$$
$$\mathsf{lookuprpt}(\mathsf{i},\mathsf{j};\mathsf{l})$$
$$\{\exists \mathsf{j}_0.\; \mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \;*\; \mathbf{dlist}\,\beta\,(\mathsf{l},\mathsf{j},\mathbf{nil},\mathsf{m})\}$$
$$\{\mathbf{dlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j}) \;*\; \mathbf{dlist}\,\beta\,(\mathsf{l},\mathsf{j},\mathbf{nil},\mathsf{m})\}.$$

Deleting an Element from a Doubly-Linked List



$\{\exists \mathsf{j}, \mathsf{l}.\ \ \mathsf{dlist}\,\alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j}) \ *\ \mathsf{k} \mapsto \mathsf{b}, \mathsf{l}, \mathsf{j} \ *\ \mathsf{dlist}\,\beta\,(\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{l} := [\mathsf{k} + 1]\ ;\ \mathsf{j} := [\mathsf{k} + 2]\ ;$

$\{\mathsf{dlist}\,\alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j}) \ *\ \mathsf{k} \mapsto \mathsf{b}, \mathsf{l}, \mathsf{j} \ *\ \mathsf{dlist}\,\beta\,(\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\mathbf{dispose}\ \mathsf{k}\ ;\ \mathbf{dispose}\ \mathsf{k} + 1\ ;\ \mathbf{dispose}\ \mathsf{k} + 2\ ;$

$\{\mathsf{dlist}\,\alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j}) \ *\ \mathsf{dlist}\,\beta\,(\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\mathbf{setrpt}(\mathsf{l}, \mathsf{j}; \mathsf{i})\ ;$

$\{\mathsf{dlist}\,\alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j}) \ *\ \mathsf{dlist}\,\beta\,(\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

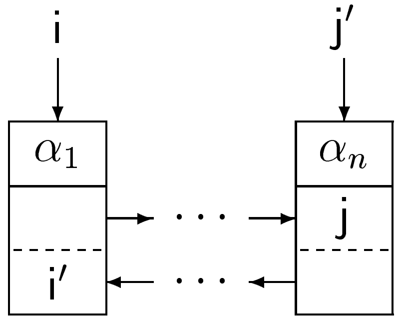
$\mathbf{setlpt}(\mathsf{l}, \mathsf{j}; \mathsf{m})$

$\{\mathsf{dlist}\,\alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j}) \ *\ \mathsf{dlist}\,\beta\,(\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\{\mathsf{dlist}\,\alpha{\cdot}\beta\,(\mathsf{i}, \mathbf{nil}, \mathbf{nil}, \mathsf{m})\}$

An Inductive Definition: Xor-Linked Lists

$\mathsf{xlist}\,\alpha\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}')$:



Definition

$$\mathsf{xlist}\,\epsilon\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') \stackrel{\mathrm{def}}{=} \mathbf{emp} \wedge \mathsf{i} = \mathsf{j} \wedge \mathsf{i}' = \mathsf{j}'$$

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

Properties

$$\mathsf{xlist}\,\mathsf{a}\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') \Leftrightarrow \mathsf{i} \mapsto \mathsf{a}, (\mathsf{j} \oplus \mathsf{i}') \wedge \mathsf{i} = \mathsf{j}'$$

$$\mathsf{xlist}\,\alpha\cdot\beta\,(\mathsf{i}, \mathsf{i}', \mathsf{k}, \mathsf{k}') \Leftrightarrow \exists \mathsf{j}, \mathsf{j}'.\ \mathsf{xlist}\,\alpha\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') \ * \ \mathsf{xlist}\,\beta\,(\mathsf{j}, \mathsf{j}', \mathsf{k}, \mathsf{k}')$$

$$\mathsf{xlist}\,\alpha\cdot\mathsf{b}\,(\mathsf{i}, \mathsf{i}', \mathsf{k}, \mathsf{k}') \Leftrightarrow \exists \mathsf{j}'.\ \mathsf{xlist}\,\alpha\,(\mathsf{i}, \mathsf{i}', \mathsf{k}', \mathsf{j}') \ * \ \mathsf{k}' \mapsto \mathsf{b}, (\mathsf{k} \oplus \mathsf{j}').$$

Emptyness Conditions (as with dlist)

$$\begin{array}{lllll}
\mathsf{xlist}\,\alpha\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') & \wedge & \mathsf{i} = \mathbf{nil} & \Rightarrow & (\alpha = \epsilon \wedge \mathsf{j} = \mathbf{nil} \wedge \mathsf{i}' = \mathsf{j}') \\
\mathsf{xlist}\,\alpha\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') & \wedge & \mathsf{j}' = \mathbf{nil} & \Rightarrow & (\alpha = \epsilon \wedge \mathsf{i}' = \mathbf{nil} \wedge \mathsf{i} = \mathsf{j}) \\
\mathsf{xlist}\,\alpha\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') & \wedge & \mathsf{i} \neq \mathsf{j} & \Rightarrow & \alpha \neq \epsilon \\
\mathsf{xlist}\,\alpha\,(\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') & \wedge & \mathsf{i}' \neq \mathsf{j}' & \Rightarrow & \alpha \neq \epsilon.
\end{array}$$

A Set-pointer Procedure for Xor-linked Lists

Since

$$
\begin{aligned}
&\{\textbf{xlist}\ \alpha\,(\mathsf{i}, \textbf{nil}, \mathsf{j}, \mathsf{j}')\}\\
&\textbf{if}\ \mathsf{j}' = \textbf{nil}\ \textbf{then}\\
&\quad \{\alpha = \epsilon \wedge \textbf{emp} \wedge \mathsf{j}' = \textbf{nil}\}\\
&\quad \mathsf{i} := \mathsf{k}\\
&\quad \{\alpha = \epsilon \wedge \textbf{emp} \wedge \mathsf{j}' = \textbf{nil} \wedge \mathsf{i} = \mathsf{k}\}\\
&\textbf{else}\\
&\quad \{\exists \alpha', \mathsf{b}, \mathsf{k}'.\ \ \alpha = \alpha' {\cdot} \mathsf{b} \wedge (\textbf{xlist}\ \alpha'\,(\mathsf{i}, \textbf{nil}, \mathsf{j}', \mathsf{k}')\ *\ \mathsf{j}' \mapsto \mathsf{b}, (\mathsf{j} \oplus \mathsf{k}'))\}\\
&\quad \textbf{newvar}\ \mathsf{x}\ \textbf{in}\ (\mathsf{x} := [\mathsf{j}' + 1]\ ;\ [\mathsf{j}' + 1] := \mathsf{x} \oplus \mathsf{j} \oplus \mathsf{k})\\
&\quad \{\exists \alpha', \mathsf{b}, \mathsf{k}'.\ \ \alpha = \alpha' {\cdot} \mathsf{b} \wedge (\textbf{xlist}\ \alpha'\,(\mathsf{i}, \textbf{nil}, \mathsf{j}', \mathsf{k}')\ *\ \mathsf{j}' \mapsto \mathsf{b}, (\mathsf{k} \oplus \mathsf{k}'))\}\\
&\{\textbf{xlist}\ \alpha\,(\mathsf{i}, \textbf{nil}, \mathsf{k}, \mathsf{j}')\},
\end{aligned}
$$

the procedure

$$
\begin{aligned}
&\mathsf{xsetrpt}(\mathsf{j}, \mathsf{j}', \mathsf{k}; \mathsf{i}) =\\
&\quad \textbf{if}\ \mathsf{j}' = \textbf{nil}\ \textbf{then}\ \mathsf{i} := \mathsf{k}\ \textbf{else}\\
&\qquad \textbf{newvar}\ \mathsf{x}\ \textbf{in}\ (\mathsf{x} := [\mathsf{j}' + 1]\ ;\ [\mathsf{j}' + 1] := \mathsf{x} \oplus \mathsf{j} \oplus \mathsf{k})
\end{aligned}
$$

satisfies

$$
\begin{aligned}
&\{\textbf{xlist}\ \alpha\,(\mathsf{i}, \textbf{nil}, \mathsf{j}, \mathsf{j}')\}\\
&\mathsf{xsetrpt}(\mathsf{j}, \mathsf{j}', \mathsf{k}; \mathsf{i})\\
&\{\textbf{xlist}\ \alpha\,(\mathsf{i}, \textbf{nil}, \mathsf{k}, \mathsf{j}')\}.
\end{aligned}
$$

Upon Reflection

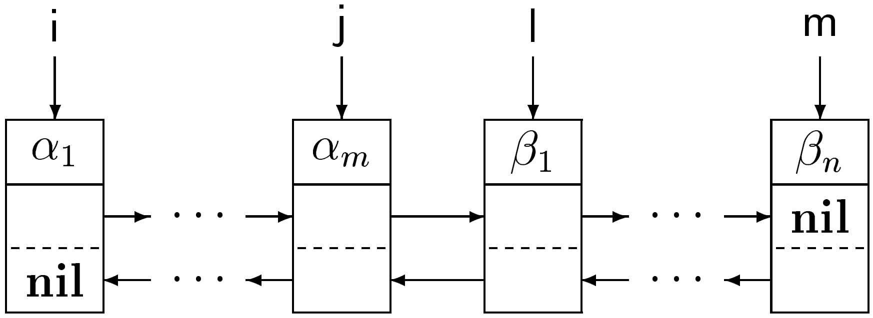
The procedure

$$\begin{aligned}
&\mathsf{xsetlpt}(\mathsf{i}, \mathsf{i}', \mathsf{k}; \mathsf{j}') = \\
&\quad \textbf{if } \mathsf{i} = \textbf{nil then } \mathsf{j}' := \mathsf{k} \textbf{ else} \\
&\qquad \textbf{newvar } \mathsf{x} \textbf{ in } (\mathsf{x} := [\mathsf{i}+1]\,;\, [\mathsf{i}+1] := \mathsf{x} \oplus \mathsf{i}' \oplus \mathsf{k})
\end{aligned}$$

satisfies

$$\begin{aligned}
&\{\textbf{xlist}\ \alpha\,(\mathsf{i}, \mathsf{i}', \textbf{nil}, \mathsf{j}')\} \\
&\mathsf{xsetlpt}(\mathsf{i}, \mathsf{i}', \mathsf{k}; \mathsf{j}') \\
&\{\textbf{xlist}\ \alpha\,(\mathsf{i}, \mathsf{k}, \textbf{nil}, \mathsf{j}')\}.
\end{aligned}$$

Inserting an Element into a Xor-Linked List



$\{\mathbf{xlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j})\ *\ \mathbf{xlist}\ \beta\,(\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{k} := \mathbf{cons}(\mathsf{a}, \mathsf{l} \oplus \mathsf{j})\ ;$

$\{\mathbf{xlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{l}, \mathsf{j})\ *\ \mathsf{k} \mapsto \mathsf{a}, (\mathsf{l} \oplus \mathsf{j})\ *\ \mathbf{xlist}\ \beta\,(\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{xsetrpt}(\mathsf{l}, \mathsf{j}, \mathsf{k}; \mathsf{i})\ ;$

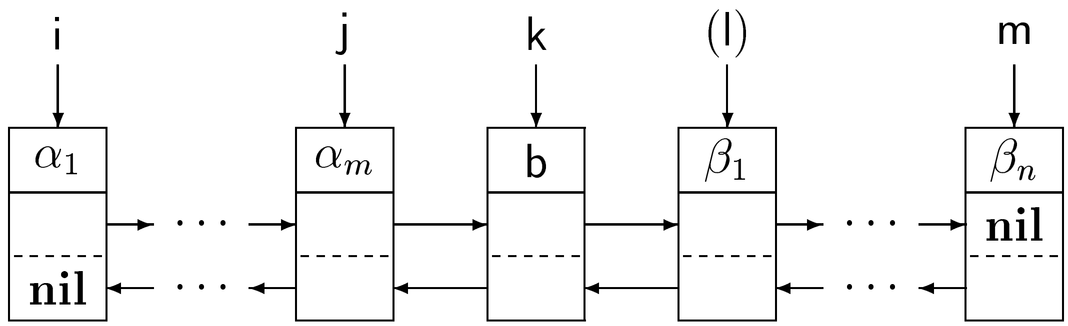
$\{\mathbf{xlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j})\ *\ \mathsf{k} \mapsto \mathsf{a}, (\mathsf{l} \oplus \mathsf{j})\ *\ \mathbf{xlist}\ \beta\,(\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

$\mathsf{xsetlpt}(\mathsf{l}, \mathsf{j}, \mathsf{k}; \mathsf{m})$

$\{\mathbf{xlist}\ \alpha\,(\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j})\ *\ \mathsf{k} \mapsto \mathsf{a}, (\mathsf{l} \oplus \mathsf{j})\ *\ \mathbf{xlist}\ \beta\,(\mathsf{l}, \mathsf{k}, \mathbf{nil}, \mathsf{m})\}$

$\{\mathbf{xlist}\ \alpha{\cdot}\mathsf{a}{\cdot}\beta\,(\mathsf{i}, \mathbf{nil}, \mathbf{nil}, \mathsf{m})\}$

Deleting an Element from a Xor-Linked List



$\{\exists \mathsf{l}.\ \ \mathsf{xlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{k},\mathsf{j})\ *\ \mathsf{k}\mapsto\mathsf{b},(\mathsf{l}\oplus\mathsf{j})\ *\ \mathsf{xlist}\,\beta\,(\mathsf{l},\mathsf{k},\mathbf{nil},\mathsf{m})\}$

$\mathsf{l} := [\mathsf{k}+1]\oplus\mathsf{j}\ ;$

$\{\mathsf{xlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{k},\mathsf{j})\ *\ \mathsf{k}\mapsto\mathsf{b},(\mathsf{l}\oplus\mathsf{j})\ *\ \mathsf{xlist}\,\beta\,(\mathsf{l},\mathsf{k},\mathbf{nil},\mathsf{m})\}$

$\mathbf{dispose}\ \mathsf{k}\ ;\ \mathbf{dispose}\ \mathsf{k}+1\ ;$

$\{\mathsf{xlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{k},\mathsf{j})\ *\ \mathsf{xlist}\,\beta\,(\mathsf{l},\mathsf{k},\mathbf{nil},\mathsf{m})\}$

$\mathbf{xsetrpt}(\mathsf{k},\mathsf{j},\mathsf{l};\mathsf{i})\ ;$

$\{\mathsf{xlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j})\ *\ \mathsf{xlist}\,\beta\,(\mathsf{l},\mathsf{k},\mathbf{nil},\mathsf{m})\}$

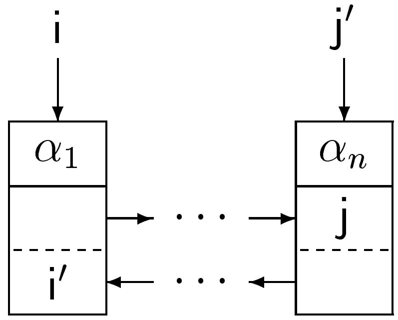
$\mathbf{xsetlpt}(\mathsf{l},\mathsf{k},\mathsf{j};\mathsf{m})$

$\{\mathsf{xlist}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j})\ *\ \mathsf{xlist}\,\beta\,(\mathsf{l},\mathsf{j},\mathbf{nil},\mathsf{m})\}$

$\{\mathsf{xlist}\,\alpha{\cdot}\beta\,(\mathsf{i},\mathbf{nil},\mathbf{nil},\mathsf{m})\}$

Reversing an Xor-Linked List

$\mathsf{xlist}\ \alpha\ (\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}')$:



$$\mathsf{xlist}\ \alpha\ (\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') \Leftrightarrow \mathsf{xlist}\ \alpha^{\dagger}\ (\mathsf{j}', \mathsf{j}, \mathsf{i}', \mathsf{i})$$

Proof (by induction on α):

Base case:

$$\mathsf{xlist}\ \epsilon\ (\mathsf{i}, \mathsf{i}', \mathsf{j}, \mathsf{j}') \Leftrightarrow \mathbf{emp} \wedge \mathsf{i} = \mathsf{j} \wedge \mathsf{i}' = \mathsf{j}' \Leftrightarrow \mathsf{xlist}\ \epsilon^{\dagger}\ (\mathsf{j}', \mathsf{j}, \mathsf{i}', \mathsf{i})$$

Induction step:

$$\begin{aligned}
&\mathsf{xlist}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i}, \mathsf{i}', \mathsf{k}, \mathsf{k}') \\
&\quad \Leftrightarrow \exists \mathsf{j}.\ \mathsf{i} \mapsto \mathsf{a}, (\mathsf{j} \oplus \mathsf{i}')\ *\ \mathsf{xlist}\ \alpha\ (\mathsf{j}, \mathsf{i}, \mathsf{k}, \mathsf{k}') \\
&\quad \Leftrightarrow \exists \mathsf{j}.\ \mathsf{xlist}\ \alpha^{\dagger}\ (\mathsf{k}', \mathsf{k}, \mathsf{i}, \mathsf{j})\ *\ \mathsf{i} \mapsto \mathsf{a}, (\mathsf{i}' \oplus \mathsf{j}) \\
&\quad \Leftrightarrow \mathsf{xlist}\ \alpha^{\dagger}{\cdot}\mathsf{a}\ (\mathsf{k}', \mathsf{k}, \mathsf{i}', \mathsf{i}) \\
&\quad \Leftrightarrow \mathsf{xlist}\ (\mathsf{a}{\cdot}\alpha)^{\dagger}\ (\mathsf{k}', \mathsf{k}, \mathsf{i}', \mathsf{i}).
\end{aligned}$$

Iterated Separating Conjunction

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

$$\begin{aligned}
&[\![\textstyle\bigodot_{v=e}^{e'} p]\!]_{\text{assert}}(s,h) = \\
&\qquad \textbf{let } m = [\![e]\!]_{\text{exp}}s,\ n = [\![e']\!]_{\text{exp}}s,\ I = \{\, i \mid m \le i \le n \,\} \textbf{ in} \\
&\qquad\quad \exists H \in I \to \text{Heaps}. \\
&\qquad\qquad \forall i, j \in I.\ i \ne j \text{ implies } Hi \perp Hj \\
&\qquad\qquad \text{and } h = \textstyle\bigcup\{\, Hi \mid i \in I \,\} \\
&\qquad\qquad \text{and } \forall i \in I.\ [\![p]\!]_{\text{assert}}([\, s \mid v\!: i \,], Hi).
\end{aligned}$$

Axiom Schemata

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

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

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

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

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

when q is pure

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

Array Allocation

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

$$\langle v := \textbf{allocate}\ e, (s, h)\rangle \rightsquigarrow ([\, s \mid v{:}\, \ell\,], h \cdot h')$$
$$\text{where } h \perp h' \text{ and } \operatorname{dom} h' = \{\, i \mid \ell \leq i < \ell + [\![e]\!]_{\text{exp}} s \,\}.$$

An Inference Rule

$$\begin{array}{l} \{p\} \\ v := \textbf{allocate}\ e \\ \{p \ast \bigodot_{i=v}^{v+e-1} i \mapsto -\}, \end{array}$$

where v does not occur free in p or e.

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

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

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

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

It is easy to show that

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

and

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

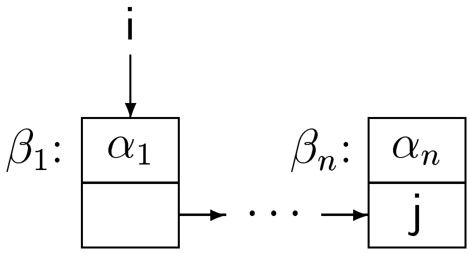
When the buffer contains a sequence α, it should satisfy

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

Inserting an Element

$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto \alpha_{k+1}) \ * \ (\bigodot_{k=0}^{n-m-1} \phi(j+k) \mapsto -))$$
$$\quad \wedge\, m = \#\alpha \wedge R \wedge m < n\}$$
$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto \alpha_{k+1}) \ * \ (\bigodot_{k=0}^{0} \phi(j+k) \mapsto -) \ *$$
$$\quad (\bigodot_{k=1}^{n-m-1} \phi(j+k) \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$
$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto \alpha_{k+1}) \ * \ \phi(j) \mapsto - \ *$$
$$\quad (\bigodot_{k=1}^{n-m-1} \phi(j+k) \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$
$$[j] := x$$
$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto \alpha_{k+1}) \ * \ \phi(j) \mapsto x \ *$$
$$\quad (\bigodot_{k=1}^{n-m-1} \phi(j+k) \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$
$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto \alpha_{k+1}) \ * \ \phi(i+m) \mapsto x \ *$$
$$\quad (\bigodot_{k=1}^{n-m-1} \phi(j+k) \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$
$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto (\alpha{\cdot}x)_{k+1}) \ * \ \phi(i+m) \mapsto (\alpha{\cdot}x)_{m+1} \ *$$
$$\quad (\bigodot_{k=1}^{n-m-1} \phi(j+k) \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$
$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto (\alpha{\cdot}x)_{k+1}) \ * \ (\bigodot_{k=m}^{m} \phi(i+k) \mapsto (\alpha{\cdot}x)_{k+1}) \ *$$
$$\quad (\bigodot_{k=1}^{n-m-1} \phi(j+k) \mapsto -)) \wedge m = \#\alpha \wedge R \wedge m < n\}$$
$$\{((\bigodot_{k=0}^{m} \phi(i+k) \mapsto (\alpha{\cdot}x)_{k+1}) \ * \ (\bigodot_{k=1}^{n-m-1} \phi(j+k) \mapsto -))$$
$$\quad \wedge\, m+1 = \#(\alpha{\cdot}x) \wedge R \wedge m < n\}$$
$$\{((\bigodot_{k=0}^{m} \phi(i+k) \mapsto (\alpha{\cdot}x)_{k+1}) \ * \ (\bigodot_{k=0}^{n-m-2} \phi(j+k+1) \mapsto -))$$
$$\quad \wedge\, m+1 = \#(\alpha{\cdot}x) \wedge R \wedge m < n\}$$
$$m := m+1\,;\, \textbf{if } j = l+n-1 \textbf{ then } j := l \textbf{ else } j := j+1$$
$$\{((\bigodot_{k=0}^{m-1} \phi(i+k) \mapsto (\alpha{\cdot}x)_{k+1}) \ * \ (\bigodot_{k=0}^{n-m-1} \phi(j+k) \mapsto -))$$
$$\quad \wedge\, m = \#(\alpha{\cdot}x) \wedge R\}$$

Connecting Two Views of Lists



If

$$\mathsf{list}\ \epsilon\ (\mathsf{i}, \mathsf{j}) \stackrel{\text{def}}{=} \mathbf{emp} \wedge \mathsf{i} = \mathsf{j}$$

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

and

$$\mathsf{listN}\ \epsilon\ (\mathsf{i}, \mathsf{j}) \stackrel{\text{def}}{=} \mathbf{emp} \wedge \mathsf{i} = \mathsf{j}$$

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

then

$$\mathsf{list}\ \alpha\ (\mathsf{i}, \mathsf{j}) \Leftrightarrow \exists \beta.\ \#\beta = \#\alpha \wedge \left(\mathsf{listN}\ \beta\ (\mathsf{i}, \mathsf{j})\ *\ \bigodot_{\mathsf{k}=1}^{\#\alpha} \beta_k \mapsto \alpha_k\right)$$

The proof is by induction on α.

Specifying Subset Lists

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

$$\{\mathsf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil})\}$$

"Set j to list of lists of subsets of i"

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

where

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

Some Properties

The predicates

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

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

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

satisfy

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

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

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

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

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

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

Proofs (1)

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

Proofs (2)

Let

$$p(\mathsf{i}) \stackrel{\text{def}}{=} \mathsf{list}\,\sigma_{\mathsf{i}}\,(\beta_{\mathsf{i}}, \mathbf{nil})$$
$$q(\mathsf{i}) \stackrel{\text{def}}{=} \gamma_{\mathsf{i}} \mapsto \mathsf{a}, (\beta^{\dagger})_{\mathsf{i}}$$
$$n \stackrel{\text{def}}{=} \#\sigma$$

Then

$$
\begin{aligned}
&Q(\sigma, \beta) \;*\; W(\beta, \gamma, \mathsf{a}) \\
&\quad \Rightarrow \Big(\#\beta = n \wedge \forall_{\mathsf{i}=1}^{\#\beta}(p(\mathsf{i}) \;*\; \mathbf{true})\Big) \;*\; \Big(\#\gamma = \#\beta \wedge \bigodot_{\mathsf{i}=1}^{\#\gamma} q(\mathsf{i})\Big) \\
&\quad \Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \Big(\Big(\forall_{\mathsf{i}=1}^{n}(p(\mathsf{i}) \;*\; \mathbf{true})\Big) \;*\; \bigodot_{\mathsf{i}=1}^{n} q(\mathsf{i})\Big) \\
&\quad \Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{\mathsf{i}=1}^{n}\Big(p(\mathsf{i}) \;*\; \mathbf{true} \;*\; \bigodot_{\mathsf{j}=1}^{n} q(\mathsf{j})\Big) \\
&\quad \Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{\mathsf{i}=1}^{n}\Big(p(\mathsf{i}) \;*\; \mathbf{true}\Big) \wedge \\
&\qquad\quad \forall_{\mathsf{i}=1}^{n}\Big(p(n+1-\mathsf{i}) \;*\; \mathbf{true} \;*\; \bigodot_{\mathsf{j}=1}^{n} q(\mathsf{j})\Big) \\
&\quad \Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{\mathsf{i}=1}^{n}\Big(p(\mathsf{i}) \;*\; \mathbf{true}\Big) \wedge \\
&\qquad\quad \forall_{\mathsf{i}=1}^{n}\Big(p(n+1-\mathsf{i}) \;*\; \mathbf{true} \;*\; q(\mathsf{i})\Big) \\
&\quad \Rightarrow \#\beta = n \wedge \#\gamma = n \wedge \forall_{\mathsf{i}=1}^{n}\Big(p(\mathsf{i}) \;*\; \mathbf{true}\Big) \wedge \\
&\qquad\quad \forall_{\mathsf{i}=1}^{n}\Big(\mathsf{list}\,(\sigma^{\dagger})_{\mathsf{i}}\,((\beta^{\dagger})_{\mathsf{i}}, \mathbf{nil}) \;*\; \mathbf{true} \;*\; \gamma_{\mathsf{i}} \mapsto \mathsf{a}, (\beta^{\dagger})_{\mathsf{i}}\Big) \\
&\quad \Rightarrow \#\gamma = n \wedge \#\beta = n \wedge \forall_{\mathsf{i}=1}^{n}\Big(\mathsf{list}\,\sigma_{\mathsf{i}}\,(\beta_{\mathsf{i}}, \mathbf{nil}) \;*\; \mathbf{true}\Big) \wedge \\
&\qquad\quad \forall_{\mathsf{i}=1}^{n}\Big(\mathsf{list}\,((\mathsf{ext}_{\mathsf{a}}\sigma)^{\dagger})_{\mathsf{i}}\,(\gamma_{\mathsf{i}}, \mathbf{nil}) \;*\; \mathbf{true}\Big) \\
&\quad \Rightarrow \#\gamma{\cdot}\beta = \#(\mathsf{ext}_{\mathsf{a}}\sigma)^{\dagger}{\cdot}\sigma \wedge \\
&\qquad\quad \forall_{\mathsf{i}=1}^{\#\gamma\cdot\beta}\Big(\mathsf{list}\,((\mathsf{ext}_{\mathsf{a}}\sigma)^{\dagger}{\cdot}\sigma)_{\mathsf{i}}\,((\gamma{\cdot}\beta)_{\mathsf{i}}, \mathbf{nil}) \;*\; \mathbf{true}\Big) \\
&\quad \Rightarrow Q((\mathsf{ext}_{\mathsf{a}}\sigma)^{\dagger}{\cdot}\sigma, \gamma{\cdot}\beta)
\end{aligned}
$$

Proofs (3)

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

From (2) and (3), we have

$$
\begin{aligned}
&(Q(\sigma, \beta) \wedge R(\beta)) \;*\; W(\beta, \gamma, \mathsf{a}) \\
&\qquad \Rightarrow (Q(\sigma, \beta) \;*\; W(\beta, \gamma, \mathsf{a})) \wedge (R(\beta) \;*\; W(\beta, \gamma, \mathsf{a})) \\
&\qquad \Rightarrow Q((\mathsf{ext_a}\sigma)^{\dagger}\cdot\sigma, \gamma\cdot\beta) \wedge R(\gamma\cdot\beta)
\end{aligned}
$$

An Iterative Program for Subset Lists

Final Assertion

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

Outer Invariant

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

Inner Invariant

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

After inner loop:

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

Iterative Subset Lists: The Main Program

$\{\mathsf{list}\,\alpha\,(\mathsf{i},\mathbf{nil})\}$

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

$\{\mathsf{list}\,\alpha\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{j} \mapsto \mathbf{nil},\mathbf{nil}\}$

$\{\mathsf{list}\,\alpha\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,[\mathbf{nil}]\,(\mathsf{j},\mathbf{nil})\}$

$\{\mathsf{ss}(\epsilon,[\epsilon]) \wedge$

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

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

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

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

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

Iterative Subset Lists: Body of the Outer Loop

$$\{\exists\alpha',\alpha'',\sigma,\beta,\mathsf{a},\mathsf{k}.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha',\sigma) \wedge$$
$$(\mathsf{i} \mapsto \mathsf{a},\mathsf{k} \ * \ \mathsf{list}\,\alpha''\,(\mathsf{k},\mathbf{nil}) \ * \ \mathsf{list}\,\beta\,(\mathsf{j},\mathbf{nil}) \ * \ (Q(\sigma,\beta){\wedge}R(\beta)))\}$$
$$\big(\mathsf{a} := [\mathsf{i}]\,;\, \mathsf{k} := [\mathsf{i}+1]\,;\, \mathbf{dispose}\ \mathsf{i}\,;\, \mathbf{dispose}\ \mathsf{i}+1\,;\, \mathsf{i} := \mathsf{k}\,;$$
$$\{\exists\alpha',\alpha'',\sigma,\beta.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha',\sigma) \wedge$$
$$(\mathsf{list}\,\alpha''\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\beta\,(\mathsf{j},\mathbf{nil}) \ * \ (Q(\sigma,\beta) \wedge R(\beta)))\}$$
$$\{\exists\alpha',\alpha'',\sigma,\beta.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha',\sigma) \wedge$$
$$(\mathsf{list}\,\alpha''\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\epsilon\,(\mathsf{j},\mathsf{j}) \ * \ \mathsf{list}\,\epsilon\,(\mathsf{j},\mathsf{j}) \ * \ \mathsf{list}\,\beta\,(\mathsf{j},\mathbf{nil}) \ *$$
$$(Q(\sigma,\epsilon{\cdot}\beta) \wedge R(\epsilon{\cdot}\beta)) \ * \ W(\epsilon,\epsilon,\mathsf{a}))\}$$
$$\mathsf{l} := \mathsf{j}\,;\, \mathsf{m} := \mathsf{j}\,;$$
$$\{\exists\alpha',\alpha'',\sigma,\beta.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha',\sigma) \wedge$$
$$(\mathsf{list}\,\alpha''\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\epsilon\,(\mathsf{l},\mathsf{j}) \ * \ \mathsf{list}\,\epsilon\,(\mathsf{j},\mathsf{m}) \ * \ \mathsf{list}\,\beta\,(\mathsf{m},\mathbf{nil}) \ *$$
$$(Q(\sigma,\epsilon{\cdot}\beta) \wedge R(\epsilon{\cdot}\beta)) \ * \ W(\epsilon,\epsilon,\mathsf{a}))\}$$
$$\{\exists\alpha',\alpha'',\sigma,\beta',\beta'',\gamma.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha',\sigma) \wedge$$
$$(\mathsf{list}\,\alpha''\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{l},\mathsf{j}) \ * \ \mathsf{list}\,\beta'\,(\mathsf{j},\mathsf{m}) \ * \ \mathsf{list}\,\beta''\,(\mathsf{m},\mathbf{nil}) \ *$$
$$(Q(\sigma,\beta'{\cdot}\beta'') \wedge R(\beta'{\cdot}\beta'')) \ * \ W(\beta',\gamma,\mathsf{a}))\}$$
$$\mathbf{while}\ \mathsf{m} \neq \mathbf{nil}\ \mathbf{do}\ \textit{Body of the Inner Loop}$$
$$\{\exists\alpha',\alpha'',\sigma,\beta',\gamma.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha',\sigma) \wedge$$
$$(\mathsf{list}\,\alpha''\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{l},\mathsf{j}) \ * \ \mathsf{list}\,\beta'\,(\mathsf{j},\mathbf{nil}) \ *$$
$$(Q(\sigma,\beta') \wedge R(\beta')) \ * \ W(\beta',\gamma,\mathsf{a}))\}$$
$$\{\exists\alpha',\alpha'',\sigma,\beta',\gamma.\ \ (\mathsf{a}{\cdot}\alpha')^{\dagger}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\mathsf{a}{\cdot}\alpha',(\mathsf{ext_a}\sigma)^{\dagger}{\cdot}\sigma) \wedge$$
$$(\mathsf{list}\,\alpha''\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\gamma{\cdot}\beta'\,(\mathsf{l},\mathbf{nil}) \ * \ (Q((\mathsf{ext_a}\sigma)^{\dagger}{\cdot}\sigma,\gamma{\cdot}\beta') \wedge R(\gamma{\cdot}\beta')))\}$$
$$\{\exists\alpha',\alpha'',\sigma,\beta.\ \ \alpha'^{\dagger}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha',\sigma) \wedge$$
$$(\mathsf{list}\,\alpha''\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\beta\,(\mathsf{l},\mathbf{nil}) \ * \ (Q(\sigma,\beta) \wedge R(\beta)))\}$$
$$\mathsf{j} := \mathsf{l}\big)$$

Iterative Subset Lists: Body of the Inner Loop

$$
\begin{array}{l}
\{\exists\alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathsf{b}, \mathsf{m}''.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha', \sigma) \wedge \\
\quad (\mathsf{list}\,\alpha''\,(\mathsf{i}, \mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{l}, \mathsf{j}) \ * \ \mathsf{list}\,\beta'\,(\mathsf{j}, \mathsf{m}) \ * \\
\qquad \mathsf{m} \mapsto \mathsf{b}, \mathsf{m}'' \ * \ \mathsf{list}\,\beta''\,(\mathsf{m}'', \mathbf{nil}) \ * \\
\qquad (Q(\sigma, \beta'{\cdot}\mathsf{b}{\cdot}\beta'') \wedge R(\beta'{\cdot}\mathsf{b}{\cdot}\beta'')) \ * \ W(\beta', \gamma, \mathsf{a}))\} \\
\big(\mathsf{b} := [\mathsf{m}]\,; \\
\{\exists\alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathsf{m}''.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha', \sigma) \wedge \\
\quad (\mathsf{list}\,\alpha''\,(\mathsf{i}, \mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{l}, \mathsf{j}) \ * \ \mathsf{list}\,\beta'\,(\mathsf{j}, \mathsf{m}) \ * \\
\qquad \mathsf{m} \mapsto \mathsf{b}, \mathsf{m}'' \ * \ \mathsf{list}\,\beta''\,(\mathsf{m}'', \mathbf{nil}) \ * \\
\qquad (Q(\sigma, \beta'{\cdot}\mathsf{b}{\cdot}\beta'') \wedge R(\beta'{\cdot}\mathsf{b}{\cdot}\beta'')) \ * \ W(\beta', \gamma, \mathsf{a}))\} \\
\mathsf{m} := [\mathsf{m} + 1]\,; \\
\{\exists\alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathsf{m}'.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha', \sigma) \wedge \\
\quad (\mathsf{list}\,\alpha''\,(\mathsf{i}, \mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{l}, \mathsf{j}) \ * \ \mathsf{list}\,\beta'\,(\mathsf{j}, \mathsf{m}') \ * \\
\qquad \mathsf{m}' \mapsto \mathsf{b}, \mathsf{m} \ * \ \mathsf{list}\,\beta''\,(\mathsf{m}, \mathbf{nil}) \ * \\
\qquad (Q(\sigma, \beta'{\cdot}\mathsf{b}{\cdot}\beta'') \wedge R(\beta'{\cdot}\mathsf{b}{\cdot}\beta'')) \ * \ W(\beta', \gamma, \mathsf{a}))\} \\
\{\exists\alpha', \alpha'', \sigma, \beta', \beta'', \gamma.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha', \sigma) \wedge \\
\quad (\mathsf{list}\,\alpha''\,(\mathsf{i}, \mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{l}, \mathsf{j}) \ * \ \mathsf{list}\,\beta'{\cdot}\mathsf{b}\,(\mathsf{j}, \mathsf{m}) \ * \\
\qquad \mathsf{list}\,\beta''\,(\mathsf{m}, \mathbf{nil}) \ * \\
\qquad (Q(\sigma, \beta'{\cdot}\mathsf{b}{\cdot}\beta'') \wedge R(\beta'{\cdot}\mathsf{b}{\cdot}\beta'')) \ * \ W(\beta', \gamma, \mathsf{a}))\} \\
\mathsf{g} := \mathbf{cons}(\mathsf{a}, \mathsf{b})\,; \\
\qquad \vdots
\end{array}
$$

$$
\begin{array}{l}
\quad\vdots \\
\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha', \sigma) \wedge \\
\quad (\mathsf{list}\,\alpha''\,(\mathsf{i}, \mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{l}, \mathsf{j}) \ * \ \mathsf{list}\,\beta'{\cdot}\mathsf{b}\,(\mathsf{j}, \mathsf{m}) \ * \\
\qquad \mathsf{list}\,\beta''\,(\mathsf{m}, \mathbf{nil}) \ * \\
\qquad (Q(\sigma, \beta'{\cdot}\mathsf{b}{\cdot}\beta'') \wedge R(\beta'{\cdot}\mathsf{b}{\cdot}\beta'')) \ * \ W(\beta', \gamma, \mathsf{a}) \ * \\
\qquad \mathsf{g} \mapsto \mathsf{a}, \mathsf{b})\} \\
\mathsf{l} := \mathbf{cons}(\mathsf{g}, \mathsf{l}) \\
\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma, \mathsf{l}'.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha', \sigma) \wedge \\
\quad (\mathsf{list}\,\alpha''\,(\mathsf{i}, \mathbf{nil}) \ * \ \mathsf{l} \mapsto \mathsf{g}, \mathsf{l}' \ * \ \mathsf{list}\,\gamma\,(\mathsf{l}', \mathsf{j}) \ * \ \mathsf{list}\,\beta'{\cdot}\mathsf{b}\,(\mathsf{j}, \mathsf{m}) \ * \\
\qquad \mathsf{list}\,\beta''\,(\mathsf{m}, \mathbf{nil}) \ * \\
\qquad (Q(\sigma, \beta'{\cdot}\mathsf{b}{\cdot}\beta'') \wedge R(\beta'{\cdot}\mathsf{b}{\cdot}\beta'')) \ * \ W(\beta', \gamma, \mathsf{a}) \ * \\
\qquad \mathsf{g} \mapsto \mathsf{a}, \mathsf{b})\} \\
\{\exists \alpha', \alpha'', \sigma, \beta', \beta'', \gamma.\ \ \alpha'^{\dagger}{\cdot}\mathsf{a}{\cdot}\alpha'' = \alpha \wedge \mathsf{ss}(\alpha', \sigma) \wedge \\
\quad (\mathsf{list}\,\alpha''\,(\mathsf{i}, \mathbf{nil}) \ * \ \mathsf{list}\,\mathsf{g}{\cdot}\gamma\,(\mathsf{l}, \mathsf{j}) \ * \ \mathsf{list}\,\beta'{\cdot}\mathsf{b}\,(\mathsf{j}, \mathsf{m}) \ * \\
\qquad \mathsf{list}\,\beta''\,(\mathsf{m}, \mathbf{nil}) \ * \\
\qquad (Q(\sigma, \beta'{\cdot}\mathsf{b}{\cdot}\beta'') \wedge R(\beta'{\cdot}\mathsf{b}{\cdot}\beta'')) \ * \ W(\beta'{\cdot}\mathsf{b}, \mathsf{g}{\cdot}\gamma, \mathsf{a}))\}\big)\,;
\end{array}
$$

A Recursive Program for Subset Lists

Let

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

which satisfies

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

Then the procedure

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

satisfies

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

since

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

Reasoning about the Call

From the recursion assumption:

$$\{\mathsf{list}\,\beta\,(\mathsf{i},\mathbf{nil})\}$$
$$\mathsf{extapp}(\mathsf{a},\mathsf{i},\mathsf{j};\mathsf{k})$$
$$\{\exists\gamma.\ \ \mathsf{list}\,\beta\,(\mathsf{i},\mathbf{nil})\ *\ \mathsf{list}\,\gamma\,(\mathsf{k},\mathsf{j})\ *\ W'(\beta,\gamma,\mathsf{a})\}$$

by free variable substitution (FVS):

$$\{\mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\}$$
$$\mathsf{extapp}(\mathsf{a},\mathsf{i}',\mathsf{j};\mathsf{k})$$
$$\{\exists\gamma.\ \ \mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\ *\ \mathsf{list}\,\gamma\,(\mathsf{k},\mathsf{j})\ *\ W'(\beta',\gamma,\mathsf{a})\}$$

by renaming (RN) in the postcondition:

$$\{\mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\}$$
$$\mathsf{extapp}(\mathsf{a},\mathsf{i}',\mathsf{j};\mathsf{k})$$
$$\{\exists\gamma'.\ \ \mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\ *\ \mathsf{list}\,\gamma'\,(\mathsf{k},\mathsf{j})\ *\ W'(\beta',\gamma',\mathsf{a})\}$$

by the frame rule (FR):

$$\{(\beta = \mathsf{b}{\cdot}\beta' \wedge \mathsf{i} \mapsto \mathsf{b},\mathsf{i}')\ *\ \mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\}$$
$$\mathsf{extapp}(\mathsf{a},\mathsf{i}',\mathsf{j};\mathsf{k})\ ;$$
$$\{(\beta = \mathsf{b}{\cdot}\beta' \wedge \mathsf{i} \mapsto \mathsf{b},\mathsf{i}')\ *$$
$$\quad(\exists\gamma'.\ \ \mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\ *\ \mathsf{list}\,\gamma'\,(\mathsf{k},\mathsf{j})\ *\ W'(\beta',\gamma',\mathsf{a}))\}$$

by introducing existential quantification (EQ):

$$\{\exists\beta'.\ \ (\beta = \mathsf{b}{\cdot}\beta' \wedge \mathsf{i} \mapsto \mathsf{b},\mathsf{i}')\ *\ \mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\}$$
$$\mathsf{extapp}(\mathsf{a},\mathsf{i}',\mathsf{j};\mathsf{k})\ ;$$
$$\{\exists\beta'.\ \ (\beta = \mathsf{b}{\cdot}\beta' \wedge \mathsf{i} \mapsto \mathsf{b},\mathsf{i}')\ *$$
$$\quad(\exists\gamma'.\ \ \mathsf{list}\,\beta'\,(\mathsf{i}',\mathbf{nil})\ *\ \mathsf{list}\,\gamma'\,(\mathsf{k},\mathsf{j})\ *\ W'(\beta',\gamma',\mathsf{a}))\}$$

Another Recursive Procedure

Let

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

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

and, as before,

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

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

$$\#\mathsf{ext}_{\mathsf{a}}\sigma \stackrel{\text{def}}{=} \#\sigma$$

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

which satisfy

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

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

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

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

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

Then the procedure

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

satisfies

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

since

$$
\begin{array}{l}
\{\mathsf{list}\,\alpha\,(\mathsf{i},\mathbf{nil})\}\\
\mathbf{if}\ \mathsf{i} = \mathbf{nil}\ \mathbf{then}\ \mathsf{j} := \mathbf{cons}(\mathbf{nil},\mathbf{nil})\ \mathbf{else}\\
\quad \{\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{nil}))\}\\
\quad \mathbf{newvar}\ \mathsf{a},\mathsf{i}',\mathsf{j}'\ \mathbf{in}\\
\quad\quad \big(\mathsf{a} := [\mathsf{i}]\,;\,\mathsf{i}' := [\mathsf{i}+1]\,;\\
\quad\quad \{\exists \alpha'.\ \ \alpha = \mathsf{a}{\cdot}\alpha' \wedge (\mathsf{i} \mapsto \mathsf{a},\mathsf{i}' \ * \ \mathsf{list}\,\alpha'\,(\mathsf{i}',\mathbf{nil}))\}\\
\quad\quad \{\exists \alpha'.\ \ (\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{i} \mapsto \mathsf{a},\mathsf{i}') \ * \ \mathsf{list}\,\alpha'\,(\mathsf{i}',\mathbf{nil})\}\\
\quad\quad \mathsf{subsets}(\mathsf{i}';\mathsf{j}')\,;\\
\quad\quad \{\exists \alpha'.\ \ (\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{i} \mapsto \mathsf{a},\mathsf{i}') \ *\\
\quad\quad\quad \big(\exists \sigma',\beta'.\ \ \mathsf{ss}'(\alpha',\sigma') \wedge \big(\mathsf{list}\,\alpha'\,(\mathsf{i}',\mathbf{nil}) \ *\\
\quad\quad\quad\quad \mathsf{list}\,\beta'\,(\mathsf{j}',\mathbf{nil}) \ * \ (Q(\sigma',\beta') \wedge R(\beta'))\big)\big)\}\\
\quad\quad \{\exists \alpha',\sigma',\beta'.\ \ \big(\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{ss}'(\alpha',\sigma') \wedge\\
\quad\quad\quad\quad (\mathsf{list}\,\mathsf{a}{\cdot}\alpha'\,(\mathsf{i},\mathbf{nil}) \ * \ (Q(\sigma',\beta') \wedge R(\beta')))\big) \ *\\
\quad\quad\quad \mathsf{list}\,\beta'\,(\mathsf{j}',\mathbf{nil})\}\\
\quad\quad \mathsf{extapp}(\mathsf{a},\mathsf{j}',\mathsf{j}';\mathsf{j})\\
\quad\quad \{\exists \alpha',\sigma',\beta'.\ \ \big(\alpha = \mathsf{a}{\cdot}\alpha' \wedge \mathsf{ss}'(\alpha',\sigma') \wedge\\
\quad\quad\quad\quad (\mathsf{list}\,\mathsf{a}{\cdot}\alpha'\,(\mathsf{i},\mathbf{nil}) \ * \ (Q(\sigma',\beta') \wedge R(\beta')))\big) \ *\\
\quad\quad\quad (\exists \gamma.\ \ \mathsf{list}\,\beta'\,(\mathsf{j}',\mathbf{nil}) \ * \ \mathsf{list}\,\gamma\,(\mathsf{j},\mathsf{j}') \ * \ W'(\beta',\gamma,\mathsf{a}))\}\\
\quad\quad \{\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\quad\quad \big(\mathsf{list}\,\mathsf{a}{\cdot}\alpha'\,(\mathsf{i},\mathbf{nil}) \ * \ \mathsf{list}\,\gamma{\cdot}\beta'\,(\mathsf{j},\mathbf{nil}) \ *\\
\quad\quad\quad\quad (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},\mathbf{nil}) \ * \ \mathsf{list}\,\beta\,(\mathsf{j},\mathbf{nil}) \ * \ (Q(\sigma,\beta) \wedge R(\beta))\big)\}
\end{array}
$$

Intuitionistic Assertions

We say that an assertion p is *intuitionistic* iff, for all states s and heaps h and h':

$$h \subseteq h' \text{ and } [\![p]\!]_{\text{assert}}(s, h) \text{ implies } [\![p]\!]_{\text{assert}}(s, h').$$

Consider

$$[\![p \ast \mathbf{true}]\!]_{\text{assert}}(s, h) = \exists h'.\ h' \subseteq h \text{ and } [\![p]\!]_{\text{assert}}(s, h').$$

It follows that

- $p \ast \mathbf{true}$ is intuitionistic.
- $p \Rightarrow (p \ast \mathbf{true})$ is valid.
- If q is intuitionistic, then $(q \ast \mathbf{true}) \Rightarrow q$ is valid.
- If $p \Rightarrow q$ is valid and q is intuitionistic, then $(p \ast \mathbf{true}) \Rightarrow q$ is valid.

Thus $p \ast \mathbf{true}$ is the strongest intuitionistic assertion weaker than p.

Intuitionistic Assertions (continued)

Consider

$$[\![\mathbf{true} \mathrel{-\!\!*} p]\!]_{\text{assert}}(s, h) = \forall h'.\ h' \supseteq h \text{ implies } [\![p]\!]_{\text{assert}}(s, h').$$

It follows that

- $\mathbf{true} \mathrel{-\!\!*} p$ is intuitionistic.
- $(\mathbf{true} \mathrel{-\!\!*} p) \Rightarrow p$ is valid.
- If q is intuitionistic, then $q \Rightarrow (\mathbf{true} \mathrel{-\!\!*} q)$ is valid.
- If $q \Rightarrow p$ is valid and q is intuitionistic, then $q \Rightarrow (\mathbf{true} \mathrel{-\!\!*} p)$ is valid.

Thus $\mathbf{true} \mathrel{-\!\!*} p$ is the weakest intuitionistic assertion stronger than p.

We define:

$$\begin{aligned} \overset{\text{i}}{\neg}\, p &\overset{\text{def}}{=} \mathbf{true} \mathrel{-\!\!*} (\neg\, p) \\ p \overset{\text{i}}{\Rightarrow} q &\overset{\text{def}}{=} \mathbf{true} \mathrel{-\!\!*} (p \Rightarrow q) \\ p \overset{\text{i}}{\Leftrightarrow} q &\overset{\text{def}}{=} \mathbf{true} \mathrel{-\!\!*} (p \Leftrightarrow q) \end{aligned}$$

Intuitionistic Assertions (continued)

If p and q are intuitionistic assertions, e and e' are expressions, and r is any assertion, then the following are intuitionistic assertions:

$$\text{Any pure assertion}$$

$$e \hookrightarrow e'$$

$$\overset{\mathrm{i}}{\neg}\, p$$

$$p \wedge q$$

$$p \vee q$$

$$p \overset{\mathrm{i}}{\Rightarrow} q$$

$$p \overset{\mathrm{i}}{\Leftrightarrow} q$$

$$\forall v.\ p$$

$$\exists v.\ p$$

$$p \ast q$$

$$p \mathrel{-\!\!\ast} q$$

$$r \ast \mathbf{true}.$$

Notice what is missing: $\mapsto$, $\mathbf{emp}$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$.

Also notice that $\overset{\mathrm{i}}{\neg}$ does not obey the law of the excluded middle. For example,

$$\mathsf{x} \hookrightarrow 13 \qquad\qquad \overset{\mathrm{i}}{\neg}\,(\mathsf{x} \hookrightarrow 13)$$

are both false when the value of x is not in the domain of the heap.

Specifications for Sorting by Merging

For sequences α and β, we define

$$\begin{aligned}
\beta \simeq \alpha &\stackrel{\text{def}}{=} \beta \text{ is a rearrangement of } \alpha\text{, i.e., } \#\beta = \#\alpha \\
&\qquad \text{and there is a permutation } f \text{ such that} \\
&\qquad \forall k.\ 1 \leq k \leq \#\alpha \text{ implies } \beta_k = \alpha_{f(k)}. \\
\mathbf{ord}\,\beta &\stackrel{\text{def}}{=} \beta \text{ is ordered in nonstrict increasing order.} \\
\mathbf{list}\,\alpha\,(e,-) &\stackrel{\text{def}}{=} \exists x.\ \mathbf{list}\,\alpha\,(e,x).
\end{aligned}$$

Then

$$\begin{aligned}
&\{\mathbf{list}\,\alpha\,(\mathsf{i},\mathsf{j}_0) \land \#\alpha = \mathsf{n} \land \mathsf{n} \geq 1\} \\
&\mathsf{mergesort}(\mathsf{n}\,;\mathsf{i},\mathsf{j}) \\
&\{\exists\beta.\ \mathbf{list}\,\beta\,(\mathsf{i},-) \land \beta \simeq \alpha \land \mathbf{ord}\,\beta \land \mathsf{j} = \mathsf{j}_0\}.
\end{aligned}$$

and

$$\begin{aligned}
&\{(\mathbf{list}\,\beta_1\,(\mathsf{i}_1,-) \ast \mathbf{list}\,\beta_2\,(\mathsf{i}_2,-)) \\
&\quad \land \mathbf{ord}\,\beta_1 \land \mathbf{ord}\,\beta_2 \land \#\beta_1 = \mathsf{n}_1 \land \#\beta_2 = \mathsf{n}_2 \land \mathsf{n}_1 \geq 1 \land \mathsf{n}_2 \geq 1\} \\
&\mathsf{merge}(\mathsf{n}_1,\mathsf{n}_2,\mathsf{i}_1,\mathsf{i}_2\,;\mathsf{i}) \\
&\{\exists\beta.\ \mathbf{list}\,\beta\,(\mathsf{i},-) \land \beta \simeq \beta_1{\cdot}\beta_2 \land \mathbf{ord}\,\beta\}.
\end{aligned}$$