Class Notes for BRICS Minicourse - Fall 2003

SEPARATION LOGIC

4. Lists and List Segments

John C. Reynolds
Department of Computer Science
Carnegie Mellon University
and
BRICS
Aarhus University

Revised November 27, 2003

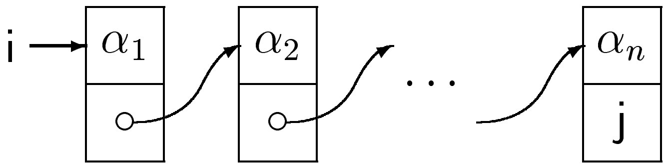
Notation for Sequences

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.
- $\alpha^\dagger$ for the reflection of α.
- $\#\alpha$ for the length of α.
- α_i for the ith component of α.

Singly-linked List Segments

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



is defined by induction on the length of the sequence α (i.e., by structural induction on α):

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

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

Properties

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

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

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

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

Emptyness Conditions

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

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

But these formulas do not say whether α is empty when $\mathsf{i} = \mathsf{j} \neq \mathbf{nil}$.

Proof of the Composition Property

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

The proof is by induction on the length of α.

When α is empty:

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

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

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

Nontouching List Segments

When

$$\mathsf{lseg}\, a_1 \cdots a_n\, (\mathsf{i}_0, \mathsf{i}_n),$$

we have

$$\begin{aligned}&\exists \mathsf{i}_1, \ldots \mathsf{i}_{n-1}.\\ &\quad (\mathsf{i}_0 \mapsto a_1, \mathsf{i}_1) \ast (\mathsf{i}_1 \mapsto a_2, \mathsf{i}_2) \ast \cdots \ast (\mathsf{i}_{n-1} \mapsto a_n, \mathsf{i}_n).\end{aligned}$$

Thus i_0, …, i_{n-1} are distinct, but i_n is not constrained, and may equal any of the i_0, …, i_{n-1}. In this case, we say that the list segment is *touching*.

We can define nontouching list segments inductively by:

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

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

or equivalently, we can define them in terms of lseg:

$$\mathsf{ntlseg}\, \alpha\, (\mathsf{i}, \mathsf{j}) \stackrel{\text{def}}{=} \mathsf{lseg}\, \alpha\, (\mathsf{i}, \mathsf{j}) \wedge \neg\, j \hookrightarrow -.$$

The obvious advantage of knowing that a list segment is nontouching is that it is easy to test whether it is empty:

$$\mathsf{ntlseg}\, \alpha\, (\mathsf{i}, \mathsf{j}) \Rightarrow (\alpha = \epsilon \Leftrightarrow \mathsf{i} = \mathsf{j}).$$

Fortunately, there are common situations where list segments must be nontouching:

$$\begin{aligned}\mathsf{list}\, \alpha\, i &\Rightarrow \mathsf{ntlseg}\, \alpha\, (\mathsf{i}, \mathbf{nil})\\ \mathsf{lseg}\, \alpha\, (i, j) \ast \mathsf{list}\, \beta\, j &\Rightarrow \mathsf{ntlseg}\, \alpha\, (i, j) \ast \mathsf{list}\, \beta\, j\\ \mathsf{lseg}\, \alpha\, (i, j) \ast \mathsf{j} \hookrightarrow - &\Rightarrow \mathsf{ntlseg}\, \alpha\, (i, j) \ast \mathsf{j} \hookrightarrow -.\end{aligned}$$

Inserting a List Element

- At the beginning:

$$
\begin{aligned}
&\{\mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\}\\
&\mathsf{k} := \mathbf{cons}(\mathsf{a},\mathsf{i});\\
&\{\mathsf{k} \mapsto \mathsf{a},\mathsf{i}\ *\ \mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\}\\
&\{\exists \mathsf{i}.\ \mathsf{k} \mapsto \mathsf{a},\mathsf{i}\ *\ \mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\}\\
&\{\mathsf{lseg}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{k},\mathsf{j})\}\\
&\mathsf{i} := \mathsf{k}\\
&\{\mathsf{lseg}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i},\mathsf{j})\}
\end{aligned}
$$

- At the end of a nonempty segment:

$$
\begin{aligned}
&\{\mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{k}\}\\
&\mathsf{l} := \mathbf{cons}(\mathsf{b},\mathsf{k})\ ;\\
&\{\mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}\\
&\{\mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a}\ *\ \mathsf{j}+1 \mapsto \mathsf{k}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}\\
&\{\mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a}\ *\ \mathsf{j}+1 \mapsto -\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}\\
&[\mathsf{j}+1] := \mathsf{l}\\
&\{\mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a}\ *\ \mathsf{j}+1 \mapsto \mathsf{l}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}\\
&\{\mathsf{lseg}\ \alpha\ (\mathsf{i},\mathsf{j})\ *\ \mathsf{j} \mapsto \mathsf{a},\mathsf{l}\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}\\
&\{\mathsf{lseg}\ \alpha{\cdot}\mathsf{a}\ (\mathsf{i},\mathsf{l})\ *\ \mathsf{l} \mapsto \mathsf{b},\mathsf{k}\}\\
&\{\mathsf{lseg}\ \alpha{\cdot}\mathsf{a}{\cdot}\mathsf{b}\ (\mathsf{i},\mathsf{k})\}
\end{aligned}
$$

Deleting a List Element

- At the beginning:

$$\{\mathsf{lseg}\ \mathsf{a}{\cdot}\alpha\,(\mathsf{i},\mathsf{k})\}$$
$$\{\exists \mathsf{j}.\ \mathsf{i} \mapsto \mathsf{a},\mathsf{j} \ * \ \mathsf{lseg}\ \alpha\,(\mathsf{j},\mathsf{k})\}$$
$$\{\mathsf{i} \mapsto \mathsf{a} \ * \ \exists \mathsf{j}.\ \mathsf{i}+1 \mapsto \mathsf{j} \ * \ \mathsf{lseg}\ \alpha\,(\mathsf{j},\mathsf{k})\}$$
$$\mathsf{j} := [\mathsf{i}+1]\ ;$$
$$\{\mathsf{i} \mapsto \mathsf{a} \ * \ \mathsf{i}+1 \mapsto \mathsf{j} \ * \ \mathsf{lseg}\ \alpha\,(\mathsf{j},\mathsf{k})\}$$
$$\textbf{dispose}\ \mathsf{i}\ ;$$
$$\{\mathsf{i}+1 \mapsto \mathsf{j} \ * \ \mathsf{lseg}\ \alpha\,(\mathsf{j},\mathsf{k})\}$$
$$\textbf{dispose}\ \mathsf{i}+1\ ;$$
$$\{\mathsf{lseg}\ \alpha\,(\mathsf{j},\mathsf{k})\}$$
$$\mathsf{i} := \mathsf{j}$$
$$\{\mathsf{lseg}\ \alpha\,(\mathsf{i},\mathsf{k})\}$$

- At the end of a nonempty segment:

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

A Cyclic Buffer

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

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

To insert an element:

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

A Cyclic Buffer (continued)

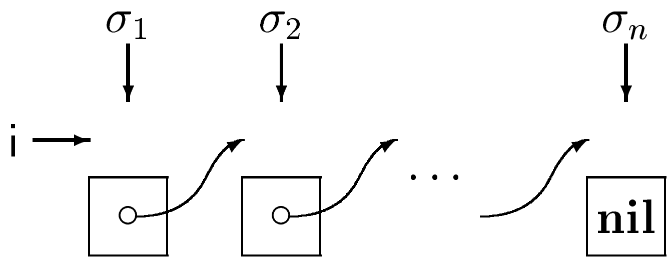
Note the above use of (LKG) for $j{:=}[j{+}1]$, with r as $((\mathsf{lseg}\ \alpha\,(\mathsf{i},\mathsf{j}') \ * \ \mathsf{j}' \mapsto \mathsf{x} \ * \ \mathsf{lseg}\ \beta\,(\mathsf{j}'',\mathsf{i})) \wedge \mathsf{m} = \#\alpha \wedge \mathsf{n} - 1 = \#\alpha + \#\beta)$.

To remove an element:

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

Bornat Lists

$\mathsf{listN}\,\sigma\,\mathsf{i}$:



is defined by

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

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

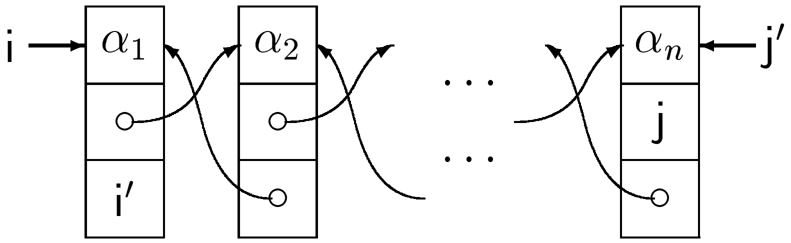
Similarly, one can define Bornat list segments and nontouching Bornat list segments.

Reversing a Bornat List

$$\{\mathsf{listN}\,\sigma_0\,\mathsf{i}\}$$
$$\{\mathsf{listN}\,\sigma_0\,\mathsf{i}\ *\ (\mathbf{emp} \wedge \mathbf{nil} = \mathbf{nil})\}$$
$$\mathsf{j} := \mathbf{nil}\,;$$
$$\{\mathsf{listN}\,\sigma_0\,\mathsf{i}\ *\ (\mathbf{emp} \wedge \mathsf{j} = \mathbf{nil})\}$$
$$\{\mathsf{listN}\,\sigma_0\,\mathsf{i}\ *\ \mathsf{listN}\,\epsilon\,\mathsf{j}\}$$
$$\{\exists \sigma, \tau.\ (\mathsf{listN}\,\sigma\,\mathsf{i}\ *\ \mathsf{listN}\,\tau\,\mathsf{j}) \wedge \sigma_0^\dagger = \sigma^\dagger{\cdot}\tau\}$$
$$\mathbf{while}\ \mathsf{i} \neq \mathbf{nil}\ \mathbf{do}$$
$$\big(\{\exists \sigma, \tau.\ (\mathsf{listN}\,(\mathsf{i}{\cdot}\sigma)\,\mathsf{i}\ *\ \mathsf{listN}\,\tau\,\mathsf{j}) \wedge \sigma_0^\dagger = (\mathsf{i}{\cdot}\sigma)^\dagger{\cdot}\tau\}$$
$$\{\exists \sigma, \tau, \mathsf{k}.\ (\mathsf{i}+1 \mapsto \mathsf{k}\ *\ \mathsf{listN}\,\sigma\,\mathsf{k}\ *\ \mathsf{listN}\,\tau\,\mathsf{j})$$
$$\wedge\ \sigma_0^\dagger = (\mathsf{i}{\cdot}\sigma)^\dagger{\cdot}\tau\}$$
$$\mathsf{k} := [\mathsf{i}+1]\,;$$
$$\{\exists \sigma, \tau.\ (\mathsf{i}+1 \mapsto \mathsf{k}\ *\ \mathsf{listN}\,\sigma\,\mathsf{k}\ *\ \mathsf{listN}\,\tau\,\mathsf{j})$$
$$\wedge\ \sigma_0^\dagger = (\mathsf{i}{\cdot}\sigma)^\dagger{\cdot}\tau\}$$
$$[\mathsf{i}+1] := \mathsf{j}\,;$$
$$\{\exists \sigma, \tau.\ (\mathsf{i}+1 \mapsto \mathsf{j}\ *\ \mathsf{listN}\,\sigma\,\mathsf{k}\ *\ \mathsf{listN}\,\tau\,\mathsf{j})$$
$$\wedge\ \sigma_0^\dagger = (\mathsf{i}{\cdot}\sigma)^\dagger{\cdot}\tau\}$$
$$\{\exists \sigma, \tau.\ (\mathsf{listN}\,\sigma\,\mathsf{k}\ *\ \mathsf{listN}\,(\mathsf{i}{\cdot}\tau)\,\mathsf{i}) \wedge \sigma_0^\dagger = \sigma^\dagger{\cdot}\mathsf{i}{\cdot}\tau\}$$
$$\{\exists \sigma, \tau.\ (\mathsf{listN}\,\sigma\,\mathsf{k}\ *\ \mathsf{listN}\,\tau\,\mathsf{i}) \wedge \sigma_0^\dagger = \sigma^\dagger{\cdot}\tau\}$$
$$\mathsf{j} := \mathsf{i}\,;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists \sigma, \tau.\ (\mathsf{listN}\,\sigma\,\mathsf{i}\ *\ \mathsf{listN}\,\tau\,\mathsf{j}) \wedge \sigma_0^\dagger = \sigma^\dagger{\cdot}\tau\}\big)$$
$$\{\exists \sigma, \tau.\ \mathsf{listN}\,\tau\,\mathsf{j} \wedge \sigma_0^\dagger = \sigma^\dagger{\cdot}\tau \wedge \sigma = \epsilon\}$$
$$\{\mathsf{listN}\,\sigma_0^\dagger\,\mathsf{j}\}$$

Doubly-Linked List Segments

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



is defined by

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

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

Properties

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

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

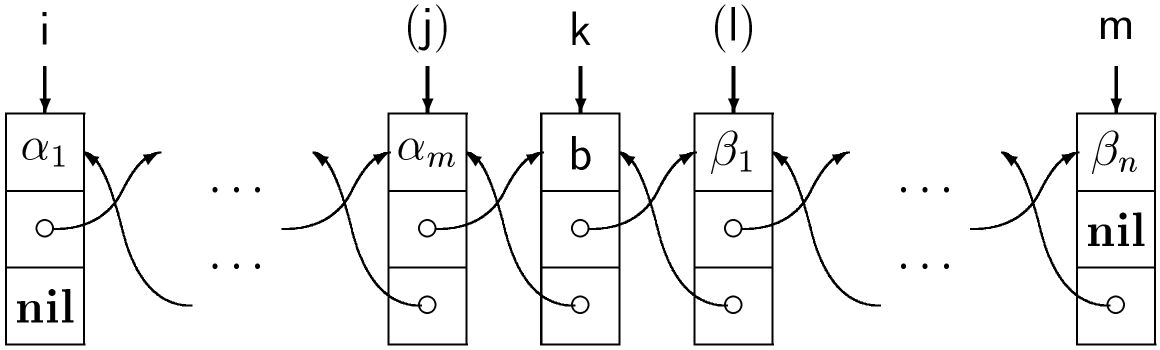
$$\mathsf{dlseg}\ \alpha{\cdot}\mathsf{b}\,(\mathsf{i}, \mathsf{i'}, \mathsf{k}, \mathsf{k'}) \Leftrightarrow \exists \mathsf{j'}.\ \mathsf{dlseg}\ \alpha\,(\mathsf{i}, \mathsf{i'}, \mathsf{k'}, \mathsf{j'}) \ *\ \mathsf{k'} \mapsto \mathsf{b}, \mathsf{k}, \mathsf{j'}.$$

Emptyness Conditions

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

(One can also define doubly-linked lists and nontouching segments.)

Deleting an Element from a Doubly-Linked List



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

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

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

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

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

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

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

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

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

$\mathbf{else}$

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

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

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

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

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

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

$\quad \vdots$

Deleting from a Doubly-Linked List (continued)

$$\vdots$$

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

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

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

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

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

$\mathbf{else}$

$$\{\exists \mathsf{a}, \beta', \mathsf{n}.\ (\mathsf{dlseg}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j})\ *\ \mathsf{l} \mapsto \mathsf{a},\mathsf{n},\mathsf{k}$$
$$*\ \mathsf{dlseg}\,\beta'\,(\mathsf{n},\mathsf{l},\mathbf{nil},\mathsf{m})) \wedge \beta = \mathsf{a}{\cdot}\beta'\}$$

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

$$\{\exists \mathsf{a}, \beta', \mathsf{n}.\ (\mathsf{dlseg}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{l},\mathsf{j})\ *\ \mathsf{l} \mapsto \mathsf{a},\mathsf{n},\mathsf{j}$$
$$*\ \mathsf{dlseg}\,\beta'\,(\mathsf{n},\mathsf{l},\mathbf{nil},\mathsf{m})) \wedge \beta = \mathsf{a}{\cdot}\beta'\}$$

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

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

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 modified 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

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

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

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

A Lookup-Pointer Procedure for Doubly-Linked Lists

Since

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

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

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

A Set-Pointer Procedure for Doubly-Linked Lists

Since

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

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

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

Upon Reflection

The procedure

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

satisfies

$$\begin{aligned}&\{\mathsf{dlseg}\,\alpha\,(\mathsf{i},\mathsf{i}'_0,\mathbf{nil},\mathsf{j}')\}\\&\mathsf{lookuplpt}(\mathsf{i},\mathsf{j}';\mathsf{i}')\\&\{\mathsf{dlseg}\,\alpha\,(\mathsf{i},\mathsf{i}'_0,\mathbf{nil},\mathsf{j}') \wedge \mathsf{i}' = \mathsf{i}'_0\},\end{aligned}$$

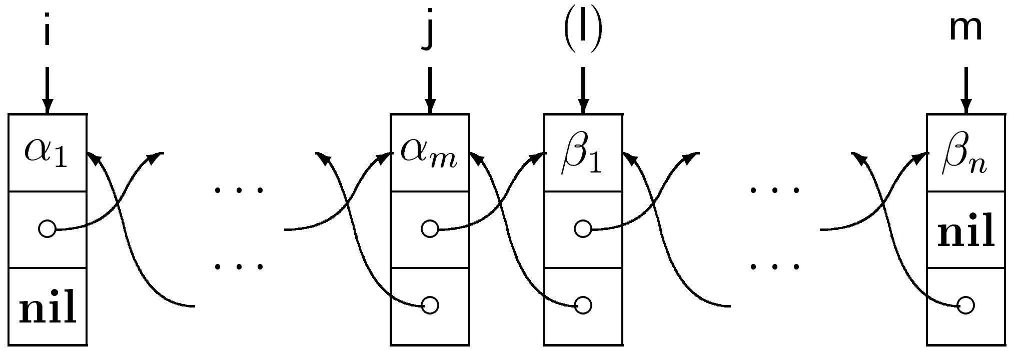
and the procedure

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

satisfies

$$\begin{aligned}&\{\exists \mathsf{i}'.\ \ \mathsf{dlseg}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathbf{nil},\mathsf{j}')\}\\&\mathsf{setlpt}(\mathsf{i},\mathsf{i}';\mathsf{j}')\\&\{\mathsf{dlseg}\,\alpha\,(\mathsf{i},\mathsf{i}',\mathbf{nil},\mathsf{j}')\}.\end{aligned}$$

Inserting an Element into a Doubly-Linked List



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

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

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

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

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

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

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

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

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

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

Deriving the Specification of the Call of lookuprpt

From:

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

using Free Variable Substitution (FVS):

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

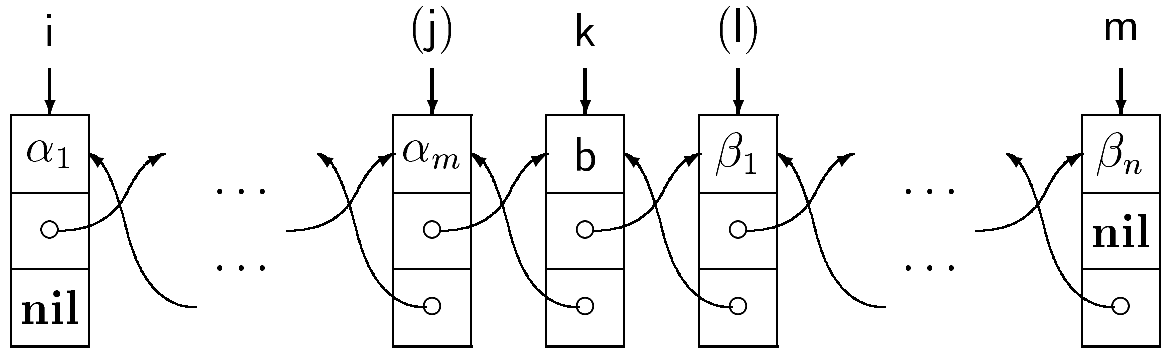
using the Frame Rule (FR) and an obvious property of equality:

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

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

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

Deleting an Element from a Doubly-Linked List



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

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

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

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

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

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

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

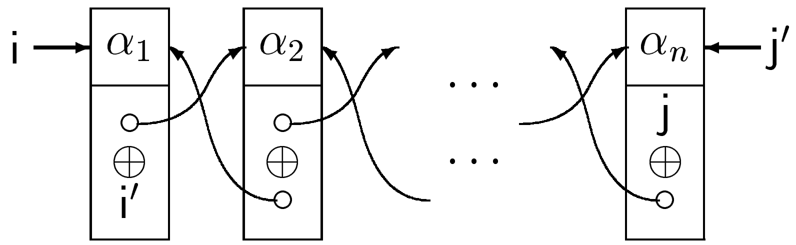
$\mathbf{setlpt}(\mathsf{l}, \mathsf{j}; \mathsf{m})$

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

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

An Inductive Definition: Xor-Linked List Segments

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



is defined by

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

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

Properties

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

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

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

Emptyness Conditions (as with dlseg)

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

A Set-Pointer Procedure for Xor-linked Lists

Since

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

the procedure

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

satisfies

$$
\begin{array}{l}
\{\mathsf{xlseg}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{j},\mathsf{j}')\}\\
\mathsf{xsetrpt}(\mathsf{j},\mathsf{j}',\mathsf{k};\mathsf{i})\\
\{\mathsf{xlseg}\,\alpha\,(\mathsf{i},\mathbf{nil},\mathsf{k},\mathsf{j}')\}.
\end{array}
$$

Upon Reflection

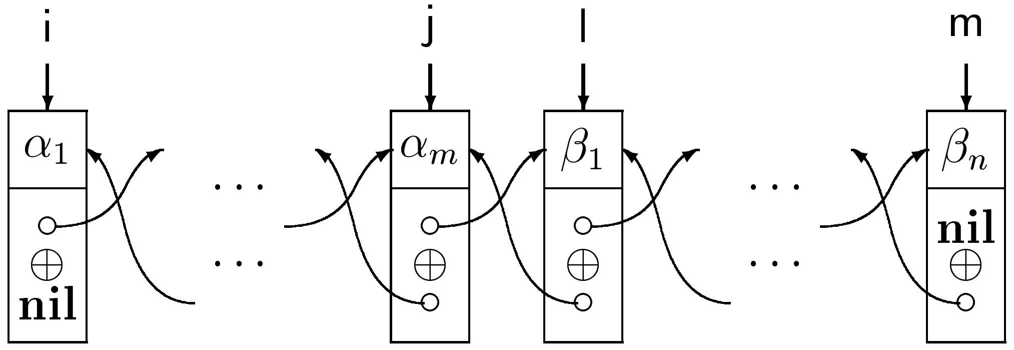
The procedure

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

satisfies

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

Inserting an Element into a Xor-Linked List



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

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

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

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

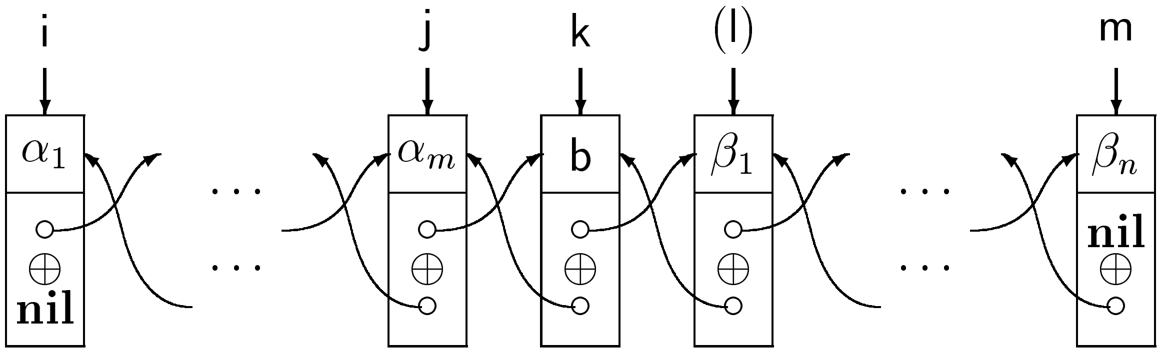
$\{\mathsf{xlseg}\ \alpha\ (\mathsf{i}, \mathbf{nil}, \mathsf{k}, \mathsf{j})\ *\ \mathsf{k} \mapsto \mathsf{a}, (\mathsf{l} \oplus \mathsf{j})\ *\ \mathsf{xlseg}\ \beta\ (\mathsf{l}, \mathsf{j}, \mathbf{nil}, \mathsf{m})\}$

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

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

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

Deleting an Element from a Xor-Linked List



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

$\mathbf{newvar}\ \mathsf{x}\ \mathbf{in}\ (\mathsf{x} := [\mathsf{k}+1]\ ; \mathsf{l} := \mathsf{x} \oplus \mathsf{j})\ ;$

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

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

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

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

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

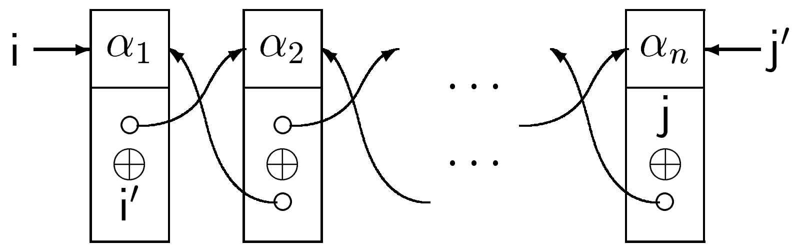
$\mathbf{xsetlpt}(\mathsf{l}, \mathsf{k}, \mathsf{j}; \mathsf{m})$

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

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

Reversing an Xor-Linked List Segment

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



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

Proof (by induction on α):

Base case:

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

Induction step:

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

Specifications for Sorting by Merging

For sequences α and β, we define

$$\beta \sim \alpha \overset{\text{def}}{=} \beta \text{ is a rearrangement of } \alpha\text{, i.e., } \#\beta = \#\alpha \text{ and there is a permutation } f \text{ such that } \forall k.\ 1 \leq k \leq \#\alpha \text{ implies } \beta_k = \alpha_{f(k)}.$$

$$\mathbf{ord}\,\beta \overset{\text{def}}{=} \beta \text{ is ordered in nonstrict increasing order.}$$

$$\mathsf{lseg}\,\alpha\,(e, -) \overset{\text{def}}{=} \exists x.\ \mathsf{lseg}\,\alpha\,(e, x).$$

Then

$$\{\mathsf{lseg}\,\alpha\,(\mathsf{i}, \mathsf{j}_0) \wedge \#\alpha = \mathsf{n} \wedge \mathsf{n} \geq 1\}$$
$$\mathsf{mergesort}(\mathsf{n}\ ;\mathsf{i}, \mathsf{j})$$
$$\{\exists\beta.\ \mathsf{lseg}\,\beta\,(\mathsf{i}, -) \wedge \beta \sim \alpha \wedge \mathbf{ord}\,\beta \wedge \mathsf{j} = \mathsf{j}_0\}.$$

and

$$\{(\mathsf{lseg}\,\beta_1\,(\mathsf{i1}, -) \wedge \mathbf{ord}\,\beta_1 \wedge \#\beta_1 = \mathsf{n1} \wedge \mathsf{n1} \geq 1)$$
$$*\ (\mathsf{lseg}\,\beta_2\,(\mathsf{i2}, -) \wedge \mathbf{ord}\,\beta_2 \wedge \#\beta_2 = \mathsf{n2} \wedge \mathsf{n2} \geq 1)\}$$
$$\mathsf{merge}(\mathsf{n1}, \mathsf{n2}, \mathsf{i1}, \mathsf{i2}\ ;\mathsf{i})$$
$$\{\exists\beta.\ \mathsf{lseg}\,\beta\,(\mathsf{i}, -) \wedge \beta \sim \beta_1{\cdot}\beta_2 \wedge \mathbf{ord}\,\beta\}.$$

A Proof for $\mathsf{mergesort}$

$\{\mathsf{lseg}\;\alpha\,(\mathsf{i},\mathsf{j}_0) \wedge \#\alpha = \mathsf{n} \wedge \mathsf{n} \geq 1\}$

if $\mathsf{n} = 1$ **then**

$\quad\{\mathsf{lseg}\;\alpha\,(\mathsf{i},-) \wedge \mathbf{ord}\;\alpha \wedge \mathsf{i} \mapsto -,\mathsf{j}_0\}$

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

$\quad\{\mathsf{lseg}\;\alpha\,(\mathsf{i},-) \wedge \mathbf{ord}\;\alpha \wedge \mathsf{j} = \mathsf{j}_0\}$

else

$\big(\mathsf{n1} := \mathsf{n} \div 2\,;\, \mathsf{n2} := \mathsf{n} - \mathsf{n1}\,;\, \mathsf{i1} := \mathsf{i}\,;$

$\{\exists\alpha_1,\alpha_2,\mathsf{i}_2.\; ((\mathsf{lseg}\;\alpha_1\,(\mathsf{i1},\mathsf{i}_2) \wedge \#\alpha_1 = \mathsf{n1} \wedge \mathsf{n1} \geq 1)$

$\quad *\;(\mathsf{lseg}\;\alpha_2\,(\mathsf{i}_2,\mathsf{j}_0) \wedge \#\alpha_2 = \mathsf{n2} \wedge \mathsf{n2} \geq 1)) \wedge \alpha = \alpha_1{\cdot}\alpha_2\}$

$\mathsf{mergesort}(\mathsf{n1};\mathsf{i1},\mathsf{i2})\,;$

$\{\exists\alpha_1,\alpha_2,\beta_1.\; ((\mathsf{lseg}\;\beta_1\,(\mathsf{i1},-) \wedge \beta_1 \sim \alpha_1 \wedge \mathbf{ord}\;\beta_1 \wedge \#\alpha_1 = \mathsf{n1} \wedge \mathsf{n1} \geq 1)$

$\quad *\;(\mathsf{lseg}\;\alpha_2\,(\mathsf{i2},\mathsf{j}_0) \wedge \#\alpha_2 = \mathsf{n2} \wedge \mathsf{n2} \geq 1)) \wedge \alpha = \alpha_1{\cdot}\alpha_2\}$

$\mathsf{mergesort}(\mathsf{n2};\mathsf{i2},\mathsf{j})\,;$

$\{\exists\alpha_1,\alpha_2,\beta_1,\beta_2.\; ((\mathsf{lseg}\;\beta_1\,(\mathsf{i1},-) \wedge \beta_1 \sim \alpha_1 \wedge \mathbf{ord}\;\beta_1 \wedge \#\alpha_1 = \mathsf{n1}$

$\qquad \wedge\, \mathsf{n1} \geq 1)$

$\quad *\;(\mathsf{lseg}\;\beta_2\,(\mathsf{i2},-) \wedge \beta_2 \sim \alpha_2 \wedge \mathbf{ord}\;\beta_2 \wedge \#\alpha_2 = \mathsf{n2} \wedge \mathsf{n2} \geq 1))$

$\quad \wedge\, \alpha = \alpha_1{\cdot}\alpha_2 \wedge \mathsf{j} = \mathsf{j}_0\}$

$\{\exists\beta_1,\beta_2.\; ((\mathsf{lseg}\;\beta_1\,(\mathsf{i1},-) \wedge \mathbf{ord}\;\beta_1 \wedge \#\beta_1 = \mathsf{n1} \wedge \mathsf{n1} \geq 1)$

$*\;(\mathsf{lseg}\;\beta_2\,(\mathsf{i2},-) \wedge \mathbf{ord}\;\beta_2 \wedge \#\beta_2 = \mathsf{n2} \wedge \mathsf{n2} \geq 1)) \wedge \alpha \sim \beta_1{\cdot}\beta_2 \wedge \mathsf{j} = \mathsf{j}_0\}$

$\mathsf{merge}(\mathsf{n1},\mathsf{n2},\mathsf{i1},\mathsf{i2};\mathsf{i})$

$\{\exists\beta,\beta_1,\beta_2.\; \mathsf{lseg}\;\beta(\mathsf{i},-) \wedge \beta \sim \beta_1{\cdot}\beta_2 \wedge \mathbf{ord}\;\beta \wedge \alpha \sim \beta_1{\cdot}\beta_2 \wedge \mathsf{j} = \mathsf{j}_0\}\big)$

$\{\exists\beta.\; \mathsf{lseg}\;\beta\,(\mathsf{i},-) \wedge \beta \sim \alpha \wedge \mathbf{ord}\;\beta \wedge \mathsf{j} = \mathsf{j}_0\}.$

A Proof for merge

$$\begin{aligned}
&\{\exists\beta, \mathsf{a1}, \mathsf{j1}, \gamma_1, \mathsf{j2}, \gamma_2. \\
&\quad (\mathsf{lseg}\,\beta\,(\mathsf{i}, \mathsf{i1}) \ * \ \mathsf{i1} \mapsto \mathsf{a1}, \mathsf{j1} \ * \ \mathsf{lseg}\,\gamma_1\,(\mathsf{j1}, -) \ * \ \mathsf{i2} \mapsto \mathsf{a2}, \mathsf{j2} \\
&\qquad * \ \mathsf{lseg}\,\gamma_2\,(\mathsf{j2}, -)) \\
&\quad \wedge \#\gamma_1 = \mathsf{n1} \wedge \#\gamma_2 = \mathsf{n2} \wedge \mathsf{a1} \leq \mathsf{a2} \wedge \beta{\cdot}\mathsf{a1}{\cdot}\gamma_1{\cdot}\mathsf{a2}{\cdot}\gamma_2 \sim \beta_1{\cdot}\beta_2 \\
&\quad \wedge \mathbf{ord}\,(\mathsf{a1}{\cdot}\gamma_1) \wedge \mathbf{ord}\,(\mathsf{a2}{\cdot}\gamma_2) \wedge \mathbf{ord}\,\beta \wedge \{\beta\} \leq^* \{\mathsf{a1}{\cdot}\gamma_1\} \cup \{\mathsf{a2}{\cdot}\gamma_2\}\} \\
&\mathsf{merge1}(\mathsf{n1}, \mathsf{n2}, \mathsf{i1}, \mathsf{i2}, \mathsf{a2}) \\
&\{\exists\beta.\ \mathsf{lseg}\,\beta\,(\mathsf{i}, -) \wedge \beta \sim \beta_1{\cdot}\beta_2 \wedge \mathbf{ord}\,\beta\}
\end{aligned}$$

Then

$$\begin{aligned}
&\{(\mathsf{lseg}\,\beta_1\,(\mathsf{i1}, -) \wedge \mathbf{ord}\,\beta_1 \wedge \#\beta_1 = \mathsf{n1} \wedge \mathsf{n1} \geq 1) \\
&\qquad * \ (\mathsf{lseg}\,\beta_2\,(\mathsf{i2}, -) \wedge \mathbf{ord}\,\beta_2 \wedge \#\beta_2 = \mathsf{n2} \wedge \mathsf{n2} \geq 1)\} \\
&\mathbf{newvar}\ \mathsf{a1}\ \mathbf{in}\ \mathbf{newvar}\ \mathsf{a2}\ \mathbf{in}\ \big(\mathsf{a1} := [\mathsf{i1}]\,;\, \mathsf{a2} := [\mathsf{i2}]\,; \\
&\quad \mathbf{if}\ \mathsf{a1} \leq \mathsf{a2}\ \mathbf{then}\ (\mathsf{i} := \mathsf{i1}\,;\, \mathsf{merge1}(\mathsf{n1}, \mathsf{n2}, \mathsf{i1}, \mathsf{i2}, \mathsf{a2})) \\
&\quad \mathbf{else}\ (\mathsf{i} := \mathsf{i2}\,;\, \mathsf{merge1}(\mathsf{n2}, \mathsf{n1}, \mathsf{i2}, \mathsf{i1}, \mathsf{a1}))\big) \\
&\{\exists\beta.\ \mathsf{lseg}\,\beta\,(\mathsf{i}, -) \wedge \beta \sim \beta_1{\cdot}\beta_2 \wedge \mathbf{ord}\,\beta\}.
\end{aligned}$$

Thus we may define

$$\begin{aligned}
&\mathsf{merge}(\mathsf{n1}, \mathsf{n2}, \mathsf{i1}, \mathsf{i2}\,;\, \mathsf{i}) = \\
&\quad \mathbf{newvar}\ \mathsf{a1}\ \mathbf{in}\ \mathbf{newvar}\ \mathsf{a2}\ \mathbf{in}\ \big(\mathsf{a1} := [\mathsf{i1}]\,;\, \mathsf{a2} := [\mathsf{i2}]\,; \\
&\qquad \mathbf{if}\ \mathsf{a1} \leq \mathsf{a2}\ \mathbf{then}\ (\mathsf{i} := \mathsf{i1}\,;\, \mathsf{merge1}(\mathsf{n1}, \mathsf{n2}, \mathsf{i1}, \mathsf{i2}, \mathsf{a2})) \\
&\qquad \mathbf{else}\ (\mathsf{i} := \mathsf{i2}\,;\, \mathsf{merge1}(\mathsf{n2}, \mathsf{n1}, \mathsf{i2}, \mathsf{i1}, \mathsf{a1}))\big)
\end{aligned}$$

A Proof for merge1

$$
\begin{array}{l}
\{\exists \beta, \mathsf{a1}, \mathsf{j1}, \gamma_1, \mathsf{j2}, \gamma_2.\\
\quad (\mathsf{lseg}\,\beta\,(\mathsf{i},\mathsf{i1}) \;*\; \mathsf{i1} \mapsto \mathsf{a1}, \mathsf{j1} \;*\; \mathsf{lseg}\,\gamma_1\,(\mathsf{j1}, -) \;*\; \mathsf{i2} \mapsto \mathsf{a2}, \mathsf{j2}\\
\qquad * \; \mathsf{lseg}\,\gamma_2\,(\mathsf{j2}, -))\\
\quad \wedge \#\gamma_1 = \mathsf{n1} \wedge \#\gamma_2 = \mathsf{n2} \wedge \mathsf{a1} \leq \mathsf{a2} \wedge \beta\cdot\mathsf{a1}\cdot\gamma_1\cdot\mathsf{a2}\cdot\gamma_2 \sim \beta_1\cdot\beta_2\\
\quad \wedge \mathbf{ord}\,(\mathsf{a1}\cdot\gamma_1) \wedge \mathbf{ord}\,(\mathsf{a2}\cdot\gamma_2) \wedge \mathbf{ord}\,\beta \wedge \{\beta\} \leq^* \{\mathsf{a1}\cdot\gamma_1\} \cup \{\mathsf{a2}\cdot\gamma_2\}\}\\
\mathbf{if}\ \mathsf{n1} = 0\ \mathbf{then}\ [\mathsf{i1} + 1] := \mathsf{i2}\\
\quad \{\exists \beta, \mathsf{a1}, \mathsf{j2}, \gamma_2.\\
\qquad (\mathsf{lseg}\,\beta\,(\mathsf{i},\mathsf{i1}) \;*\; \mathsf{i1} \mapsto \mathsf{a1}, \mathsf{i2} \;*\; \mathsf{i2} \mapsto \mathsf{a2}, \mathsf{j2} \;*\; \mathsf{lseg}\,\gamma_2\,(\mathsf{j2}, -))\\
\qquad \wedge \#\gamma_2 = \mathsf{n2} \wedge \mathsf{a1} \leq \mathsf{a2} \wedge \beta\cdot\mathsf{a1}\cdot\mathsf{a2}\cdot\gamma_2 \sim \beta_1\cdot\beta_2\\
\qquad \wedge \mathbf{ord}\,(\mathsf{a2}\cdot\gamma_2) \wedge \mathbf{ord}\,\beta \wedge \{\beta\} \leq^* \{\mathsf{a1}\} \cup \{\mathsf{a2}\cdot\gamma_2\}\}\\
\mathbf{else\ newvar}\ \mathsf{j1}\ \mathbf{in\ newvar}\ \mathsf{a3}\ \mathbf{in}\\
\quad \big(\mathsf{j1} := [\mathsf{i1} + 1]\,;\, \mathsf{a3} := [\mathsf{j1}]\,;\\
\quad \{\exists \beta, \mathsf{a1}, \mathsf{j3}, \gamma_3, \mathsf{j2}, \gamma_2.\\
\qquad (\mathsf{lseg}\,\beta\,(\mathsf{i},\mathsf{i1}) \;*\; \mathsf{i1} \mapsto \mathsf{a1}, \mathsf{j1} \;*\; \mathsf{j1} \mapsto \mathsf{a3}, \mathsf{j3} \;*\; \mathsf{lseg}\,\gamma_3\,(\mathsf{j3}, -)\\
\qquad\quad * \; \mathsf{i2} \mapsto \mathsf{a2}, \mathsf{j2} \;*\; \mathsf{lseg}\,\gamma_2\,(\mathsf{j2}, -))\\
\quad \wedge \#\gamma_3 = \mathsf{n1} - 1 \wedge \#\gamma_2 = \mathsf{n2} \wedge \mathsf{a1} \leq \mathsf{a2} \wedge \beta\cdot\mathsf{a1}\cdot\mathsf{a3}\cdot\gamma_3\cdot\mathsf{a2}\cdot\gamma_2 \sim \beta_1\cdot\beta_2\\
\quad \wedge \mathbf{ord}\,(\mathsf{a1}\cdot\mathsf{a3}\cdot\gamma_3) \wedge \mathbf{ord}\,(\mathsf{a2}\cdot\gamma_2) \wedge \mathbf{ord}\,\beta\\
\quad \wedge \{\beta\} \leq^* \{\mathsf{a1}\cdot\mathsf{a3}\cdot\gamma_3\} \cup \{\mathsf{a2}\cdot\gamma_2\}\}\\
\quad \mathbf{if}\ \mathsf{a3} \leq \mathsf{a2}\ \mathbf{then}\ \mathsf{merge1}(\mathsf{n1} - 1, \mathsf{n2}, \mathsf{j1}, \mathsf{i2}, \mathsf{a2})\\
\quad \mathbf{else}\ ([\mathsf{i1} + 1] := \mathsf{i2}\,;\, \mathsf{merge1}(\mathsf{n2}, \mathsf{n1} - 1, \mathsf{i2}, \mathsf{j1}, \mathsf{a3})\big)\\
\{\exists \beta.\ \mathsf{lseg}\,\beta\,(\mathsf{i}, -) \wedge \beta \sim \beta_1\cdot\beta_2 \wedge \mathbf{ord}\,\beta\}
\end{array}
$$