Overview of Separation Logic

--From a beginner's perspective

He Xiao

Department of Computer Science, UIUC

2015-10-30

Motivation of Separation Logic

- Shared mutable data structures are prevalent in modern programming languages
  - Aliasing
  - Address arithmetics

- Standard Hoare Logic does not support these features

Motivation Cont.

```
int *a = new int;

*a = 5;

int *b = a;

*b = 22;

cout << "a is " << *a << endl;

cout << "b is " << *b << endl;
```

Motivation Cont.

$\{*a = 5 \wedge *b = 5\}$

*b := 22;

$\{*a = 22 \wedge *b = 22\}$

NOT a Hoare Triple

Separation Logic came to rescue

- Extend Hoare Logic in three aspects to handle shared mutable data structures
  - simple imperative language
  - assertion language
  - specification

Overall structure of S.L. specification

Program

{Pre} C {Post}

Assertions

Extended Imperative P.L.

$$
\begin{array}{lr}
\langle\text{comm}\rangle ::= \cdots & \\
\quad \mid \langle\text{var}\rangle := \mathbf{cons}(\langle\text{exp}\rangle, \ldots, \langle\text{exp}\rangle) & \text{allocation} \\
\quad \mid \langle\text{var}\rangle := [\langle\text{exp}\rangle] & \text{lookup} \\
\quad \mid [\langle\text{exp}\rangle] := \langle\text{exp}\rangle & \text{mutation} \\
\quad \mid \mathbf{dispose}\,\langle\text{exp}\rangle & \text{deallocation}
\end{array}
$$

$$\text{Values} = \text{Integers}$$

$$\text{Atoms} \cup \text{Addresses} \subseteq \text{Integers}$$

where Atoms and Addresses are disjoint

$$\text{Heaps} = \bigcup_{A \overset{\text{fin}}{\subseteq} \text{Addresses}} (A \to \text{Values}).$$

$$\mathbf{nil} \in \text{Atoms}$$

$$\text{Stores}_V = V \to \text{Values}$$

$$\text{States}_V = \text{Stores}_V \times \text{Heaps},$$

$$[\![e \in \langle\text{exp}\rangle]\!]_{\text{exp}} \in \Big(\bigcup_{V \overset{\text{fin}}{\supseteq} \text{FV}(e)} \text{Stores}_V\Big) \to \text{Values}$$

$$[\![b \in \langle\text{boolexp}\rangle]\!]_{\text{bexp}} \in \Big(\bigcup_{V \overset{\text{fin}}{\supseteq} \text{FV}(b)} \text{Stores}_V\Big) \to \{\mathbf{true}, \mathbf{false}\}$$

Extended Imperative P.L. (Cont.)

- semantics for the new commands
  - Program configuration:

    Non-terminal : <c,(s, h)>

    Terminal : (s, h) or abort

- γ ~>* γ' : finite sequence of transitions
- $\gamma\uparrow$: infinite sequence of transitions from γ

Extended Imperative P.L. (Cont.)

- Allocation

$$\frac{}{\begin{array}{l}\langle v := \mathbf{cons}(e_1, \ldots, e_n), (s, h)\rangle \\ \quad \rightsquigarrow ([\, s \mid v{:}\, \ell\,], [\, h \mid \ell{:}\, [\![e_1]\!]_{\exp} s \mid \ldots \mid \ell{+}n{-}1{:}\, [\![e_n]\!]_{\exp} s\,]),\end{array}}$$

where $\ell, \ldots, \ell + n - 1 \in \text{Addresses} - \operatorname{dom} h$.

- Lookup

When $[\![e]\!]_{\exp} s \in \operatorname{dom} h$:

$$\frac{}{\langle v := [e], (s, h)\rangle \rightsquigarrow ([\, s \mid v{:}\, h([\![e]\!]_{\exp} s)\,], h),}$$

When $[\![e]\!]_{\exp} s \notin \operatorname{dom} h$:

$$\frac{}{\langle v := [e], (s, h)\rangle \rightsquigarrow \mathbf{abort}.}$$

- Mutation

When $[\![e]\!]_{\exp} s \in \operatorname{dom} h$:

$$\frac{}{\langle [e] := e', (s, h)\rangle \rightsquigarrow (s, [\, h \mid [\![e]\!]_{\exp} s{:}\, [\![e']\!]_{\exp} s\,]),}$$

When $[\![e]\!]_{\exp} s \notin \operatorname{dom} h$:

$$\frac{}{\langle [e] := e', (s, h)\rangle \rightsquigarrow \mathbf{abort}.}$$

- Deallocation

When $[\![e]\!]_{\exp} s \in \operatorname{dom} h$:

$$\frac{}{\langle \mathbf{dispose}\; e, (s, h)\rangle \rightsquigarrow (s, h\rceil(\operatorname{dom} h - \{[\![e]\!]_{\exp} s\})),}$$

When $[\![e]\!]_{\exp} s \notin \operatorname{dom} h$:

$$\frac{}{\langle \mathbf{dispose}\; e, (s, h)\rangle \rightsquigarrow \mathbf{abort}.}$$

Extended Assertion Languages: syntax

$$
\begin{array}{lll}
\langle \text{assert} \rangle ::= & \cdots & \\
& \mid \ \mathbf{emp} & \text{empty heap} \\
& \mid \ \langle \text{exp} \rangle \mapsto \langle \text{exp} \rangle & \text{singleton heap} \\
& \mid \ \langle \text{assert} \rangle \ * \ \langle \text{assert} \rangle & \text{separating conjunction} \\
& \mid \ \langle \text{assert} \rangle \mathrel{-\!\!*} \langle \text{assert} \rangle & \text{separating implication}
\end{array}
$$

Extended Assertion Languages: Semantics

$$[\![p \in \langle\text{assert}\rangle]\!]_{\text{asrt}} \in$$
$$(\bigcup_{V \overset{\text{fin}}{\supseteq} \text{FV}(p)} \text{Stores}_V) \to \text{Heaps} \to \{\textbf{true}, \textbf{false}\}.$$

Specifically, **emp** asserts that the heap is empty:

$$[\![\textbf{emp}]\!]_{\text{asrt}} s\, h \text{ iff } \operatorname{dom} h = \{\},$$

$e \mapsto e'$ asserts that the heap contains one cell, at address e with contents e':

$$[\![e \mapsto e']\!]_{\text{asrt}} s\, h \text{ iff}$$
$$\operatorname{dom} h = \{[\![e]\!]_{\exp} s\} \text{ and } h([\![e]\!]_{\exp} s) = [\![e']\!]_{\exp} s,$$

Extended Assertion Languages: Separating Conjunction

$$
\begin{aligned}
&[\![p_0 \; * \; p_1]\!]_{\text{asrt}} s\, h \;\text{ iff} \\
&\qquad \exists h_0, h_1.\; h_0 \perp h_1 \text{ and } h_0 \cdot h_1 = h \text{ and} \\
&\qquad\qquad [\![p_0]\!]_{\text{asrt}} s\, h_0 \text{ and } [\![p_1]\!]_{\text{asrt}} s\, h_1,
\end{aligned}
$$

Extended Assertion Languages: separating implication

$$
\begin{aligned}
&[\![p_0 -\!\!* \; p_1]\!]_{\mathrm{asrt}}\, s\, h \;\text{ iff} \\
&\qquad \forall h'.\ (h' \perp h \text{ and } [\![p_0]\!]_{\mathrm{asrt}}\, s\, h') \text{ implies} \\
&\qquad\qquad [\![p_1]\!]_{\mathrm{asrt}}\, s\, (h \cdot h').
\end{aligned}
$$

Extended Assertion Languages (Cont.)

$$e \mapsto - \stackrel{\text{def}}{=} \exists x'.\, e \mapsto x' \quad \text{where } x' \text{ not free in } e$$

$$e \hookrightarrow e' \stackrel{\text{def}}{=} e \mapsto e' \, * \, \mathbf{true}$$

$$e \mapsto e_1, \ldots, e_n$$
$$\stackrel{\text{def}}{=} e \mapsto e_1 \, * \, \cdots \, * \, e+n-1 \mapsto e_n$$

$$e \hookrightarrow e_1, \ldots, e_n$$
$$\stackrel{\text{def}}{=} e \hookrightarrow e_1 \, * \, \cdots \, * \, e+n-1 \hookrightarrow e_n$$
$$\text{iff } e \mapsto e_1, \ldots, e_n \, * \, \mathbf{true}.$$

Similar program specification

$$\begin{array}{llr} \langle\text{spec}\rangle ::= & \{\langle\text{assert}\rangle\}\ \langle\text{comm}\rangle\ \{\langle\text{assert}\rangle\} & \text{partial} \\ & \mid [\ \langle\text{assert}\rangle\ ]\ \langle\text{comm}\rangle\ [\ \langle\text{assert}\rangle\ ] & \text{total} \end{array}$$

Let $V = \mathrm{FV}(p) \cup \mathrm{FV}(c) \cup \mathrm{FV}(q)$. Then

$$\begin{aligned} &\{p\}\ c\ \{q\} \text{ holds iff } \forall(s,h) \in \mathrm{States}_V.\ [\![p]\!]_{\mathrm{asrt}} s\, h \text{ implies} \\ &\qquad \neg\,(c, (s,h) \rightsquigarrow^* \mathbf{abort}) \\ &\qquad \text{and } (\forall(s',h') \in \mathrm{States}_V. \\ &\qquad\qquad c, (s,h) \rightsquigarrow^* (s',h') \text{ implies } [\![q]\!]_{\mathrm{asrt}} s'\, h'), \end{aligned}$$

and

$$\begin{aligned} &[\ p\ ]\ c\ [\ q\ ] \text{ holds iff } \forall(s,h) \in \mathrm{States}_V.\ [\![p]\!]_{\mathrm{asrt}} s\, h \text{ implies} \\ &\qquad \neg\,(c, (s,h) \rightsquigarrow^* \mathbf{abort}) \\ &\qquad \text{and } \neg\,(c, (s,h) \uparrow) \\ &\qquad \text{and } (\forall(s',h') \in \mathrm{States}_V. \\ &\qquad\qquad c, (s,h) \rightsquigarrow^* (s',h') \text{ implies } [\![q]\!]_{\mathrm{asrt}} s'\, h'). \end{aligned}$$

program specification (Cont.)

- Most inference rules of Hoare Logic remain sound

- One exception: rule of constancy

$$\frac{\{p\}\ c\ \{q\}}{\{p \wedge r\}\ c\ \{q \wedge r\}},$$

$$\frac{\{\mathsf{x} \mapsto -\}\ [\mathsf{x}] := 4\ \{\mathsf{x} \mapsto 4\}}{\{\mathsf{x} \mapsto - \wedge \mathsf{y} \mapsto 3\}\ [\mathsf{x}] := 4\ \{\mathsf{x} \mapsto 4 \wedge \mathsf{y} \mapsto 3\}}$$

program specification (Cont.)

- Frame Rule solves the scalability problem

- Frame Rule

$$\frac{\{p\}\ c\ \{q\}}{\{p * r\}\ c\ \{q * r\},}$$

Back to the previous problem

```
//Translated to the simple imperative language
a := cons(5); //alloc
b := a; //normal assignment
{a -> 5 /\ b -> 5}
[b] := 22; // mutation
{a -> 22 /\ b -> 22}
```

Non aliasing is also OK

```
int data = 5;

int *a = new int;

*a = data;

int *b = new int;

*b = data;

*b = 22;
```

```
//Separation Logic Spec
{emp}
data := 5;
a := cons(data);
b := cons(data);
{a -> 5 * b -> 5}
[b] := 22;
{a -> 5 * b -> 22}
```

Verify list-reversing program

$$\mathsf{j} := \mathbf{nil}\,;\,\mathbf{while}\ \mathsf{i} \neq \mathbf{nil}\ \mathbf{do}$$
$$(\mathsf{k} := [\mathsf{i}+1]\,;\,[\mathsf{i}+1] := \mathsf{j}\,;\,\mathsf{j} := \mathsf{i}\,;\,\mathsf{i} := \mathsf{k}).$$

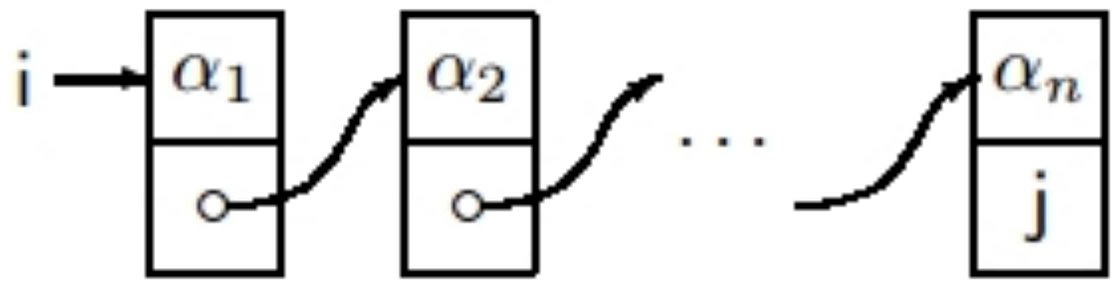
Notations in List Reasoning

- 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.

List reasoning (Cont.)

- A simple graphical representation of

  list α (i,j)



$$\{\exists\alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil}) \ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta \wedge \mathsf{i} \neq \mathbf{nil}\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i}, \mathbf{nil}) \ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta, \mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k} \ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil}) \ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i} + 1]\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k} \ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil}) \ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i} + 1] := \mathsf{j}\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{j} \ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil}) \ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil}) \ *\ \mathbf{list}\ \mathsf{a}{\cdot}\beta\ (\mathsf{i}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists\alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil}) \ *\ \mathbf{list}\ \beta\ (\mathsf{i}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists\alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil}) \ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}.$$

$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta \wedge \mathsf{i} \neq \mathbf{nil}\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta, \mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i} + 1]\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i} + 1] := \mathsf{j}\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{j}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \mathsf{a}{\cdot}\beta\ (\mathsf{i}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{i}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}.$$

$\{\exists \alpha, \beta.\ (\textbf{list}\ \alpha\ (\mathsf{i}, \textbf{nil})\ *\ \textbf{list}\ \beta\ (\mathsf{j}, \textbf{nil}))$

$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger \cdot \beta \wedge \mathsf{i} \neq \textbf{nil}\}$

$\{\exists \mathsf{a}, \alpha, \beta.\ (\textbf{list}\ \mathsf{a}\cdot\alpha\ (\mathsf{i}, \textbf{nil})\ *\ \textbf{list}\ \beta\ (\mathsf{j}, \textbf{nil}))$

$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}\cdot\alpha)^\dagger \cdot \beta\}$

$\{\exists \mathsf{a}, \alpha, \beta, \mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \textbf{list}\ \alpha\ (\mathsf{k}, \textbf{nil})\ *\ \textbf{list}\ \beta\ (\mathsf{j}, \textbf{nil}))$

$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}\cdot\alpha)^\dagger \cdot \beta\}$

$\mathsf{k} := [\mathsf{i} + 1]\ ;$

$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \textbf{list}\ \alpha\ (\mathsf{k}, \textbf{nil})\ *\ \textbf{list}\ \beta\ (\mathsf{j}, \textbf{nil}))$

$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}\cdot\alpha)^\dagger \cdot \beta\}$

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

$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{j}\ *\ \textbf{list}\ \alpha\ (\mathsf{k}, \textbf{nil})\ *\ \textbf{list}\ \beta\ (\mathsf{j}, \textbf{nil}))$

$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}\cdot\alpha)^\dagger \cdot \beta\}$

$\{\exists \mathsf{a}, \alpha, \beta.\ (\textbf{list}\ \alpha\ (\mathsf{k}, \textbf{nil})\ *\ \textbf{list}\ \mathsf{a}\cdot\beta\ (\mathsf{i}, \textbf{nil}))$

$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger \cdot \mathsf{a} \cdot \beta\}$

$\{\exists \alpha, \beta.\ (\textbf{list}\ \alpha\ (\mathsf{k}, \textbf{nil})\ *\ \textbf{list}\ \beta\ (\mathsf{i}, \textbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta\}$

$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$

$\{\exists \alpha, \beta.\ (\textbf{list}\ \alpha\ (\mathsf{i}, \textbf{nil})\ *\ \textbf{list}\ \beta\ (\mathsf{j}, \textbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta\}.$

$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta \wedge \mathsf{i} \neq \mathbf{nil}\}$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\ \mathsf{a}{\cdot}\alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta,\mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\ \alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i}+1]\ ;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\ \alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i}+1] := \mathsf{j}\ ;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{j}\ *\ \mathbf{list}\ \alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\ \alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \mathsf{a}{\cdot}\beta\,(\mathsf{i},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\,(\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\,(\mathsf{i},\mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\,(\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\,(\mathsf{j},\mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}.$$

$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta \wedge \mathsf{i} \neq \mathbf{nil}\}$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta,\mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i}+1]\ ;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i}+1] := \mathsf{j}\ ;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{j}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \mathsf{a}{\cdot}\beta\ (\mathsf{i},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{i},\mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}.$$

$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta \wedge \mathsf{i} \neq \mathbf{nil}\}$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta,\mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i}+1]\ ;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i}+1] := \mathsf{j}\ ;$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathsf{i} \mapsto \mathsf{a},\mathsf{j}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists\mathsf{a},\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \mathsf{a}{\cdot}\beta\ (\mathsf{i},\mathbf{nil}))$$
$$\quad\wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{i},\mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists\alpha,\beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i},\mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j},\mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}.$$

$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta \wedge \mathsf{i} \neq \mathbf{nil}\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta, \mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i} + 1]\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i} + 1] := \mathsf{j}\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{j}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \mathsf{a}{\cdot}\beta\ (\mathsf{i}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{i}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}.$$

$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta \wedge \mathsf{i} \neq \mathbf{nil}\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \mathsf{a}{\cdot}\alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta, \mathsf{k}.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\mathsf{k} := [\mathsf{i} + 1]\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{k}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$[\mathsf{i} + 1] := \mathsf{j}\ ;$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathsf{i} \mapsto \mathsf{a}, \mathsf{j}\ *\ \mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = (\mathsf{a}{\cdot}\alpha)^\dagger{\cdot}\beta\}$$
$$\{\exists \mathsf{a}, \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \mathsf{a}{\cdot}\beta\ (\mathsf{i}, \mathbf{nil}))$$
$$\quad \wedge\ \alpha_0^\dagger = \alpha^\dagger{\cdot}\mathsf{a}{\cdot}\beta\}$$
$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{k}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{i}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}$$
$$\mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}$$
$$\{\exists \alpha, \beta.\ (\mathbf{list}\ \alpha\ (\mathsf{i}, \mathbf{nil})\ *\ \mathbf{list}\ \beta\ (\mathsf{j}, \mathbf{nil})) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta\}.$$

By Applying Consequence rule and Composition rule

$\{LI \wedge guard\}\ LB\ \{LI\}$

Related Work

- Matching Logic (Grigore Rosu, RTA'15)

– separation logic is a particular matching logic theory

– Separation logic formulae can be encoded in

matching logic and get solution.

References

- John C. Reynolds* Separation Logic: A Logic for Shared Mutable Data Structures IEEE 2002
- Grigore Rosu Matching Logic --- Extended Abstract RTA'15

Questions?