Reasoning about Iterators with Separation Logic

Neelakantan R. Krishnaswami

November 8, 2006

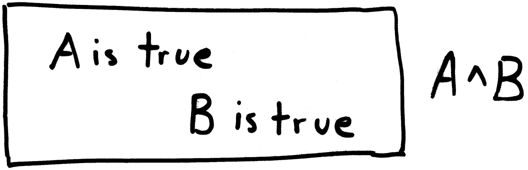
Overview

- Multiple iterators traversing a collection in parallel
- Safe changes to the collection (e.g. caching) OK; only *logical* state needs to be immutable
- Can separately check client and implementation for conformance to abstract interface

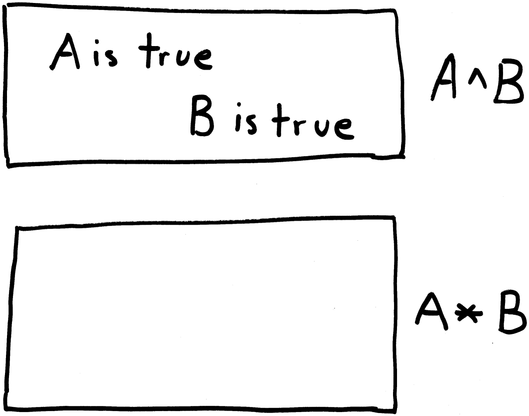
Overview

- Multiple iterators traversing a collection in parallel
- Safe changes to the collection (e.g. caching) OK; only *logical* state needs to be immutable
- Can separately check client and implementation for conformance to abstract interface
- Specification language developed in collaboration with John Reynolds, Jonathan Aldrich, and Lars Birkedal

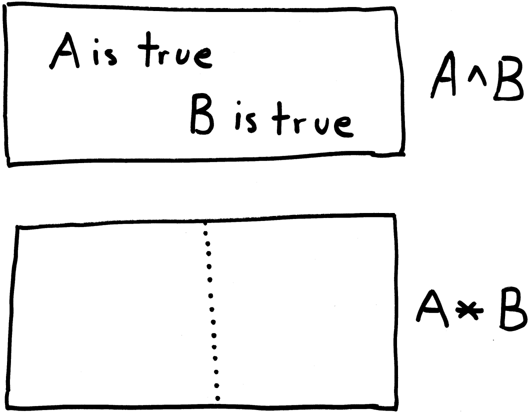
Conjunction, Regular and Separating



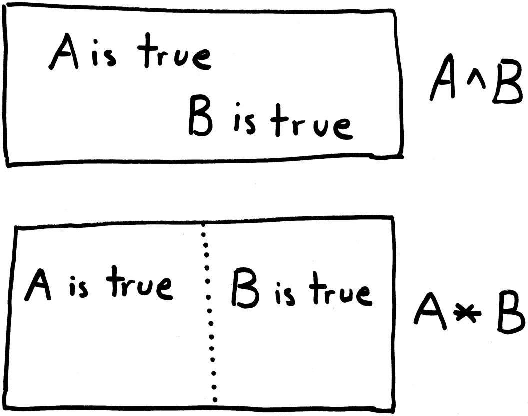
Conjunction, Regular and Separating



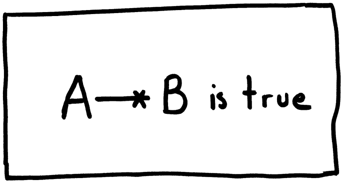
Conjunction, Regular and Separating



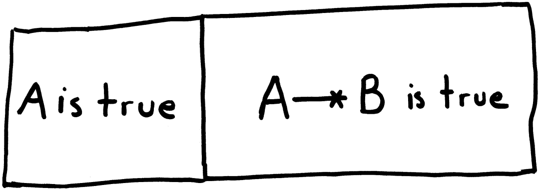
Conjunction, Regular and Separating



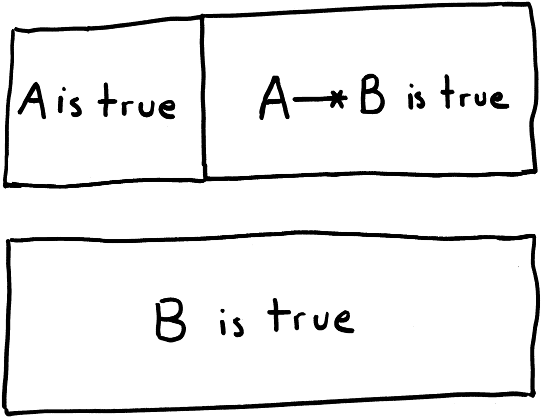
Separating Conjunction



Separating Conjunction



Separating Conjunction



The Iterator Protocol, In Separation Logic

$$\exists coll : (\tau_c \times \text{seq} \;\; \times \text{prop}) \Rightarrow \text{prop}.$$

$\{\top\}$ `new_coll()` $\{a : \tau_c.\ \exists P.\ coll(a, [], P)\}$ and

The Iterator Protocol, In Separation Logic

$\exists coll : (\tau_c \times \text{seq} \quad \times \text{prop}) \Rightarrow \text{prop}.$

$\{\top\}$ `new_coll()` $\{a : \tau_c.\ \exists P.\ coll(a, [], P)\}$ and

$\forall P, c, xs.\ \{coll(c, xs, P)\}$
$\quad \texttt{empty}(c)$
$\quad \{a : \text{bool}.\ coll(c, xs, P)\}$ and

The Iterator Protocol, In Separation Logic

$\exists coll : (\tau_c \times \text{seq} \ \times \text{prop}) \Rightarrow \text{prop}.$

$\{\top\}\ \texttt{new\_coll()}\ \{a : \tau_c.\ \exists P.\ coll(a, [], P)\}$ and

$\forall P, c, xs.\ \{coll(c, xs, P)\}$
$\texttt{empty}(c)$
$\{a : \text{bool}.\ coll(c, xs, P)\}$ and

$\forall P, c, x, xs.\ \{coll(c, xs, P)\}$
$\texttt{add}(c, x)$
$\{a : 1.\ \exists P'.\ coll(c, x :: xs, P')\}$ and

The Iterator Protocol, In Separation Logic

$$\exists iter : (\tau_i \times \tau_c \times \text{seq} \quad \times \text{prop}) \Rightarrow \text{prop}.$$

$$\begin{array}{l} \forall c, xs, P.\ \{coll(c, xs, P)\} \\ \qquad\qquad \texttt{new\_iter}(c) \\ \qquad\qquad \{a : \tau_i.\ iter(a, c, xs, P)\} \ \text{ and} \end{array}$$

The Iterator Protocol, In Separation Logic

$$\exists iter : (\tau_i \times \tau_c \times \text{seq} \quad \times \text{prop}) \Rightarrow \text{prop}.$$

$$\forall c, xs, P.\ \{coll(c, xs, P)\}$$
$$\texttt{new\_iter}(c)$$
$$\{a : \tau_i.\ iter(a, c, xs, P)\} \text{ and}$$

$$\forall i, c, xs, P.\ \{iter(i, c, xs, P)\}$$
$$\texttt{next}(i)$$
$$\{a : 1 + \text{nat}.\ iter(i, c, xs, P)\} \text{ and}$$

The Iterator Protocol, In Separation Logic

$\exists iter : (\tau_i \times \tau_c \times \text{seq} \;\; \times \text{prop}) \Rightarrow \text{prop}.$

$$\begin{aligned} \forall c, xs, P.\ & \{coll(c, xs, P)\} \\ & \texttt{new\_iter}(c) \\ & \{a : \tau_i.\ iter(a, c, xs, P)\} \text{ and} \end{aligned}$$

$$\begin{aligned} \forall i, c, xs, P.\ & \{iter(i, c, xs, P)\} \\ & \texttt{next}(i) \\ & \{a : 1 + \text{nat}.\ iter(i, c, xs, P)\} \text{ and} \end{aligned}$$

$$\begin{aligned} \forall i, c, xs, P.\ \{iter(i, c, xs, P) \supset\ & coll(c, xs, P) * \\ & coll(c, xs, P) \mathrel{-\!\!*} iter(i, c, xs, P)\} \end{aligned}$$

A Client Program

$$\{coll(c, xs, P)\}$$

A Client Program

$\{coll(c, xs, P)\}$

let $b = \texttt{empty}(c)$;

A Client Program

$\{coll(c, xs, P)\}$
let $b = \texttt{empty}(c)$;
$\{coll(c, xs)\}$

A Client Program

$\{coll(c, xs, P)\}$
let $b = \texttt{empty}(c)$;
$\{coll(c, xs)\}$
let $i_1 = \texttt{new\_iter}(c)$;

A Client Program

$\{coll(c, xs, P)\}$
$\text{let } b = \texttt{empty}(c);$
$\{coll(c, xs)\}$
$\text{let } i_1 = \texttt{new\_iter}(c);$
$\{iter(i_1, c, xs, P)\}$

A Client Program

$\{coll(c, xs, P)\}$
let b = `empty`(c);
$\{coll(c, xs)\}$
let i_1 = `new_iter`(c);
$\{iter(i_1, c, xs, P)\}$
$\{(coll(c, xs, P) * (coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))\}$

A Client Program

$\{coll(c, xs, P)\}$

$\text{let } b = \texttt{empty}(c);$

$\{coll(c, xs)\}$

$\text{let } i_1 = \texttt{new\_iter}(c);$

$\{iter(i_1, c, xs, P)\}$

$\{(coll(c, xs, P) * (coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))\}$

$\text{let } i_2 = \texttt{new\_iter}(c);$

A Client Program

$\{coll(c, xs, P)\}$

$\text{let } b = \texttt{empty}(c);$

$\{coll(c, xs)\}$

$\text{let } i_1 = \texttt{new\_iter}(c);$

$\{iter(i_1, c, xs, P)\}$

$\{(coll(c, xs, P) * (coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

$\text{let } i_2 = \texttt{new\_iter}(c);$

$\{iter(i_2, c, xs, P) * (coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

A Client Program

$$\{coll(c, xs, P)\}$$
$$\text{let } b = \texttt{empty}(c);$$
$$\{coll(c, xs)\}$$
$$\text{let } i_1 = \texttt{new\_iter}(c);$$
$$\{iter(i_1, c, xs, P)\}$$
$$\{(coll(c, xs, P) * (coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$$
$$\text{let } i_2 = \texttt{new\_iter}(c);$$
$$\{iter(i_2, c, xs, P) * (coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$$
$$\{coll(c, xs, P) * \\ (coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) * \\ (coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$$

A Client Program

1 $\{coll(c, xs, P)\}$

2 $\texttt{let } b = \texttt{empty}(c);$

3 $\{coll(c, xs)\}$

4 $\texttt{let } i_1 = \texttt{new\_iter}(c);$

5 $\{iter(i_1, c, xs, P)\}$

6 $\{(coll(c, xs, P) * (coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

7 $\texttt{let } i_2 = \texttt{new\_iter}(c);$

8 $\{iter(i_2, c, xs, P) * (coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

9 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

10 $\texttt{let } b' = \texttt{empty}(c);$

A Client Program

1 $\{coll(c, xs, P)\}$

2 $\texttt{let } b = \texttt{empty}(c);$

3 $\{coll(c, xs)\}$

4 $\texttt{let } i_1 = \texttt{new\_iter}(c);$

5 $\{iter(i_1, c, xs, P)\}$

6 $\{(coll(c, xs, P) * (coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))\}$

7 $\texttt{let } i_2 = \texttt{new\_iter}(c);$

8 $\{iter(i_2, c, xs, P) * (coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))\}$

9 $\{coll(c, xs, P)*$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))*$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

10 $\texttt{let } b' = \texttt{empty}(c);$

11 $\{coll(c, xs, P)*$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))*$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

A Client Program, Continued

$$
\begin{aligned}
11\ \{&coll(c, xs, P) *\\
&(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *\\
&(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}
\end{aligned}
$$

A Client Program, Continued

11 $\{coll(c, xs, P) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

12 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

A Client Program, Continued

11 $\{coll(c, xs, P) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

12 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

13 let $v = \texttt{next}(i_1)$;

A Client Program, Continued

11 $\{coll(c, xs, P) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

12 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

13 let $v = \texttt{next}(i_1)$;

14 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

A Client Program, Continued

11 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \multimap iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \multimap iter(i_2, c, xs, P))\}$

12 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \multimap iter(i_2, c, xs, P))\}$

13 let $v = \texttt{next}(i_1)$;

14 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \multimap iter(i_2, c, xs, P))\}$

15 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \multimap iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \multimap iter(i_2, c, xs, P))\}$

A Client Program, Continued

11 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

12 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

13 let $v = \texttt{next}(i_1)$;

14 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

15 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

16 $\{iter(i_2, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

A Client Program, Continued

11 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

12 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

13 let $v = \mathtt{next}(i_1)$;

14 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

15 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

16 $\{iter(i_2, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

17 let $v = \mathtt{next}(i_2)$;

A Client Program, Continued

11 $\{coll(c, xs, P) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

12 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

13 let $v = \texttt{next}(i_1);$

14 $\{iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

15 $\{coll(c, xs, P) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

16 $\{iter(i_2, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))\}$

17 let $v = \texttt{next}(i_2);$

18 $\{iter(i_2, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))\}$

18 $\{iter(i_2, c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

A Client Program, Continued Again

$$18\ \{iter(i_2, c, xs, P) *$$
$$(coll(c, xs, P) -\!* \ iter(i_1, c, xs, P))\}$$

$$19\ \{coll(c, xs, P) *$$
$$(coll(c, xs, P) -\!* \ iter(i_1, c, xs, P)) *$$
$$(coll(c, xs, P) -\!* \ iter(i_2, c, xs, P))\}$$

A Client Program, Continued Again

18 $\{iter(i_2, c, xs, P) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P))\}$

19 $\{coll(c, xs, P) *$
$(coll(c, xs, P) -\!\!* \; iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) -\!\!* \; iter(i_2, c, xs, P))\}$

20 $\mathtt{add}(c, x)$

A Client Program, Continued Again

18 $\{iter(i_2, c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P))\}$

19 $\{coll(c, xs, P) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

20 $\texttt{add}(c, x)$

21 $\{\exists Q.\ coll(c, xs, Q) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_1, c, xs, P)) *$
$(coll(c, xs, P) \mathrel{-\!\!*} iter(i_2, c, xs, P))\}$

Questions?

Any questions?

Implementing the Module – Invariants

$$\exists coll : (\tau_c \times \text{seq} \;\; \times \text{prop}) \Rightarrow \text{prop}.$$

Implementing the Module – Invariants

$\exists coll : (\tau_c \times \text{seq} \;\; \times \text{prop}) \Rightarrow \text{prop}.$

$coll(c, xs, P) \equiv \exists n.\ \text{snd}\ \ c \hookrightarrow n * (linked\_list(\text{fst}\ \ c, xs) \wedge P)$

Implementing the Module – Invariants

$$\exists coll : (\tau_c \times \text{seq} \ \times \text{prop}) \Rightarrow \text{prop}.$$

$$coll(c, xs, P) \equiv \exists n.\ \text{snd}\ c \hookrightarrow n * (linked\_list(\text{fst}\ c, xs) \wedge P)$$

$$\begin{array}{ll} linked\_list(c, x :: xs) & \equiv \exists c'.\ c \hookrightarrow \text{cons}(x, c') * linked\_list(c', xs) \\ linked\_list(c, []) & \equiv c \hookrightarrow \text{nil} \end{array}$$

Iterator Invariants

$$\exists iter : (\tau_i \times \tau_c \times \mathrm{seq} \quad \times \mathrm{prop}) \Rightarrow \mathrm{prop}.$$

Iterator Invariants

$$\exists iter : (\tau_i \times \tau_c \times \mathsf{seq} \ \times \mathsf{prop}) \Rightarrow \mathsf{prop}.$$

$$\begin{aligned} iter(i, c, xs, P) \equiv \exists l, n, xs_1, xs_2.& \\ & (P \wedge (seg(\mathsf{fst}\ c, l, xs_1) \ * coll(l, xs_2))) * \\ & i \hookrightarrow l \ * \mathsf{snd}\ c \hookrightarrow n \wedge \\ & xs = xs_1 \cdot xs_2 \end{aligned}$$

Iterator Invariants

$$\exists iter : (\tau_i \times \tau_c \times \text{seq} \;\; \times \text{prop}) \Rightarrow \text{prop}.$$

$$\begin{aligned}
iter(i, c, xs, P) \equiv \exists l, n, xs_1, xs_2. & \\
& (P \wedge (seg(\text{fst } c, l, xs_1) \; * \; coll(l, xs_2))) \; * \\
& i \hookrightarrow l \; * \text{snd } c \hookrightarrow n \; \wedge \\
& xs = xs_1 \cdot xs_2 \\
seg(l, l', x :: xs) \equiv \exists l''. \; l \hookrightarrow \text{cons}(x, l'') * seg(l'', xs) & \\
seg(l, l', []) \qquad \equiv l = l' &
\end{aligned}$$