Exersices of

Lecture Notes on Iris: High-Order Concurrent Separation Logic

Frank King

June 28, 2022

Contents

3 The Logics of Resources **1**
3.2 Rules for separating conjunction and magic wand 1

4 Separation Logic for Sequential Programs **2**
4.1 Derived rules for Hoare triples 7
4.2 Reasoning about Mutable Data Structures 11
4.3 Abstract Data Types 19

5 Case Study: foldr **19**

3 The Logics of Resources

3.2 Rules for separating conjunction and magic wand

Exercise 3.2. Following the example above derive the following two rules:

$$\frac{x \notin \mathrm{FV}(P)}{P * \exists x.\Phi \dashv\vdash \exists x.P * \Phi} \qquad \frac{x \notin \mathrm{FV}(P)}{P \wedge \exists x.\Phi \dashv\vdash \exists x.P \wedge \Phi}$$

where FV stands for *free variables*.

Proof. Let's prove these two rules one by one, and direction by direction.

1. The direction from left to right:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : \tau \mid P * \Phi \vdash P * \Phi \qquad \Gamma, x : \tau \vdash x : \tau \qquad x \notin \mathrm{FV}(P)}{\Gamma, x : \tau \mid P * \Phi \vdash \exists x : \tau. P * \Phi}\,(\exists I)}{\Gamma, x : \tau \mid \Phi \vdash P \mathrel{-\!\!*} \exists x : \tau. P * \Phi}\,(\mathrel{-\!\!*} I)}{\Gamma \mid \exists x : \tau. \Phi \vdash P \mathrel{-\!\!*} \exists x : \tau. P * \Phi}\,(\exists E)}{\Gamma \mid P * \exists x : \tau. \Phi \vdash \exists x : \tau. P * \Phi}\,(\mathrel{-\!\!*} E).$$

The direction from right to left:

$$\dfrac{\dfrac{\dfrac{\Gamma, x : \tau \mid \Phi \vdash \Phi \qquad \Gamma, x : \tau \vdash x : \tau}{\Gamma, x : \tau \mid \Phi \vdash \exists x : \tau.\Phi}\;(\exists\mathrm{I})}{\Gamma, x : \tau \mid P * \Phi \vdash P * \exists x : \tau.\Phi}\;(*\text{-}\textsc{mono})}{\Gamma \mid \exists x : \tau.P * \Phi \vdash P * \exists x : \tau.\Phi}\;(\exists\mathrm{E})\ .$$

2. To prove the direction from left to right, we first prove a lemma

$$\overset{\Rightarrow\mathrm{E}'}{\dfrac{Q \vdash P \Rightarrow R}{P \wedge Q \vdash R}}\,.$$

It can be derived by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{P \wedge Q \vdash P \wedge Q}\;(\textsc{Asm})}{P \wedge Q \vdash Q}\;(\wedge\mathrm{ER}) \qquad Q \vdash P \Rightarrow R}{P \wedge Q \vdash P \Rightarrow R}\;(\textsc{Cut}) \qquad \dfrac{\dfrac{}{P \wedge Q \vdash P \wedge Q}\;(\textsc{Asm})}{P \wedge Q \vdash P}\;(\wedge\mathrm{EL})}{P \wedge Q \vdash R}\;(\Rightarrow\mathrm{E})\ .$$

Then the original rule can be derived as

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : \tau \mid P \wedge \Phi \vdash P \wedge \Phi \qquad \Gamma, x : \tau \vdash x : \tau \qquad x \notin \mathrm{FV}(P)}{\Gamma, x : \tau \mid P \wedge \Phi \vdash \exists x : \tau.P \wedge \Phi}\;(\exists\mathrm{I})}{\Gamma, x : \tau \mid \Phi \vdash P \Rightarrow \exists x : \tau.P \wedge \Phi}\;(\Rightarrow\mathrm{I})}{\Gamma \mid \exists x : \tau.\Phi \vdash P \Rightarrow \exists x : \tau.P \wedge \Phi}\;(\exists\mathrm{E})}{\Gamma \mid P \wedge \exists x : \tau.\Phi \vdash \exists x : \tau.P \wedge \Phi}\;(\Rightarrow\mathrm{E}')\ .$$

The direction from right to left:

$$\dfrac{\dfrac{\Gamma, x : \tau \mid P \wedge \Phi \vdash P\ (\wedge\mathrm{EL}) \qquad \dfrac{\Gamma, x : \tau \mid P \wedge \Phi \vdash \Phi\ (\wedge\mathrm{ER}) \qquad \Gamma, x : \tau \vdash x : \tau}{\Gamma, x : \tau \mid P \wedge \Phi \vdash \exists x : \tau.\Phi}\;(\exists\mathrm{I})}{\Gamma, x : \tau \mid P \wedge \Phi \vdash P \wedge \exists x : \tau.\Phi}\;(\wedge\mathrm{I})}{\Gamma \mid \exists x : \tau.P \wedge \Phi \vdash P \wedge \exists x : \tau.\Phi}\;(\exists\mathrm{E})\ .$$

$\square$

4 Separation Logic for Sequential Programs

Exercise 4.1. The rule

$$\overset{\textsc{Ht-frame-invalid}}{\dfrac{S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P \wedge R\}\, e\, \{v.Q \wedge R\}}}$$

is not sound.

Come up with a counterexample to the above rule.

Proof. We have

$$S \vdash \{x \hookrightarrow 0 \wedge y \hookrightarrow 1\}\, y \leftarrow !\, x\, \{v.v = () \wedge y \hookrightarrow 0\},$$

but not

$$S \vdash \{x \hookrightarrow 0 \wedge y \hookrightarrow 1 \wedge y \hookrightarrow 1\}\, y \leftarrow !\, x\, \{v.v = () \wedge y \mapsto 0 \wedge y \hookrightarrow 1\}.$$

□

Exercise 4.2. Prove

$$S \vdash (\{P\}\, e\, \{v.Q\} \Rightarrow \forall R : iProp, \{P * R\}\, e\, \{v.Q * R\}).$$

Proof. It can be derived by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, R : iProp \mid S \wedge \{P\}\, e\, \{v.Q\} \vdash \{P\}\, e\, \{v.Q\}}\ (\wedge\mathrm{ER})}{\Gamma, R : iProp \mid S \wedge \{P\}\, e\, \{v.Q\} \vdash \{P * R\}\, e\, \{v.Q * R\}}\ (\textsc{Ht-frame})}{\Gamma \mid S \wedge \{P\}\, e\, \{v.Q\} \vdash \forall R : iProp, \{P * R\}\, e\, \{v.Q * R\}}\ (\forall\mathrm{I})}{\Gamma \mid S \vdash (\{P\}\, e\, \{v.Q\} \Rightarrow \forall R : iProp, \{P * R\}\, e\, \{v.Q * R\})}\ (\Rightarrow\mathrm{I})\ .$$

□

Exercise 4.3. Use Ht-bind to show $\{\mathsf{True}\}\, 3 + 4 + 5\, \{v.v = 12\}$.

Proof. It is obvious that the primary evaluation context of $3 + 4 + 5$ is $3 + 4$. We must show $\{\mathsf{True}\}\, 3 + 4\, \{w.Q\}$ and $\forall w.\{Q\}\, w + 5\, \{w.w = 12\}$ for some Q.

Let $Q \equiv w = 7$, then these two proposition is immediate to prove. □

Exercise 4.4. Prove the derived rule $P * Q \vdash P \wedge Q$ for any propositions P and Q.

Proof. It can be derived by

$$\dfrac{\dfrac{}{P * Q \vdash P}\ (*\text{-weak}) \qquad \dfrac{}{P * Q \vdash Q}\ (*\text{-weak},\ *\text{-comm})}{P * Q \vdash P \wedge Q}\ (\wedge\mathrm{I})\ .$$

□

Exercise 4.5. Use the intuitive reading of Hoare triples to explain why the above rules are sound.

Ht-disj

$$\dfrac{S \vdash \{P\}\, e\, \{v.R\} \qquad S \vdash \{Q\}\, e\, \{v.R\}}{S \vdash \{P \vee Q\}\, e\, \{v.R\}}$$

Ht-exist

$$\dfrac{x \notin Q \qquad S \vdash \forall x.\{P\}\, e\, \{v.Q\}}{x \notin Q \qquad S \vdash \{\exists x.P\}\, e\, \{v.Q\}}$$

Proof. Firstly, consider rule Ht-disj, the direction from top to bottom.

The hypothses tell us, that for any precondition P, after execution of e, then $v.R$ holds, and so for any precondition Q. It is reasonable that for any precondition $P \vee Q$, after execution of e, then $v.R$ holds.

Then the opposite direction.

The hypothsis tells us, that for any precondition $P \vee Q$, after execution of e, then $v.R$ holds. In the precondition, $P \vee Q$ (i.e., either P or Q holds) proves the postcondition $v.R$, then separately, both of P and Q as a precondition in alone, should prove the postcondition $v.R$.

Secondly, consider rule Ht-exist, the direction from top to bottom.

The hypothsis tells us, that for all x, for any precondition P, after execution of e, then $v.Q$ holds. We peek any x where P holds, then for this particular x, the precondition P must prove the postcondition $v.Q$. So $\exists x.P$, as a precondition, should prove the same postcondition, too.

Then the opposite direction.

Admitted. (Need more strict definition of Hoare triple).

Exercise 4.8. Prove the following derived rule. For any *value* u and expression e we have

$$\frac{S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P\}\, \pi_1(e, u)\, \{v.Q\}}$$

It is important that the second component is a value. Show that the following rule is not valid in general if e_1 is allowed to be an arbitrary expression.

$$\frac{S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P\}\, \pi_1(e, e_1)\, \{v.Q\}}$$

Hint: What if e and e_1 read or write to the same location?

The problem is that we know nothing about the behaviour of e_1. But we can specify its behaviour using Hoare triples. Come up with some propositions P_1 and P_2 or some conditions on P_1 and P_2 such that the following rule

$$\frac{S \vdash \{P\}\, e\, \{v.Q\} \qquad S \vdash \{P_1\}\, e_1\, \{v.P_2\}}{S \vdash \{P\}\, \pi_1(e, e_1)\, \{v.Q\}}$$

is derivable.

Proof. 1. First, we prove the derivation by Ht-bind.

$$\frac{S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P\}\, \pi_1(e, u)\, \{v.Q\}}$$

Applying Ht-bind to the goal $S \vdash \{P\}\, \pi_1(e, u)\, \{v.Q\}$, we have to show

$$S \vdash \{P\}\, (e, u)\, \{v_1.Q_1\}, \tag{1}$$
$$S \vdash \{Q_1[u_1/v_1]\}\, \pi_1 u_1\, \{v.Q\}, \tag{2}$$

for some Q_1.

Applying Ht-bind again to (1), we have to show

$$S \vdash \{P\}\, e\, \{v_2.Q_2\}, \tag{3}$$
$$S \vdash \{Q_2[u_2/v_2]\}\, (u_2, u)\, \{v_1.Q_1\}, \tag{4}$$

for some Q_2.

From the hypothesis $S \vdash \{P\}\, e\, \{v.Q\}$, (3) is immediate for $Q_2 \equiv Q[v_2/v]$. Then (4) becomes

$$S \vdash \{Q[u_2/v]\}\, (u_2, u)\, \{v_1.Q_1\}. \tag{4a}$$

Since (u_2, u) is a value, we have

$$S \vdash \{\mathsf{True}\}\, (u_2, u)\, \{v_1.v_1 = (u_2, u)\},$$

by Ht-ret, which derives

$$S \vdash \{\mathsf{True} * Q[u_2/v]\}\, (u_2, u)\, \{v_1.v_1 = (u_2, u) * Q[u_2/v]\}, \tag{4b}$$

by Ht-frame. Hence, (4a) is reached for $Q_1 \equiv v_1 = (u_2, u) * Q[u_2/v]$.

Now we only have to show the remained (2), which becomes

$$S \vdash \{u_1 = (u_2, u) * Q[u_2/v]\}\, \pi_1 u_1\, \{v.Q\}. \tag{2a}$$

The persistent proposition $u_1 = (u_2, u)$ can be moved out of the Hoare precondition, and substitute the Hoare triple with it, we have

$$S \vdash \{Q[u_2/v]\}\, \pi_1(u_2, u)\, \{v.Q\}. \tag{2b}$$

Applying Ht-proj, we have

$$S \vdash \{\mathsf{True}\}\, \pi_1(u_2, u)\, \{v.v = u_2\}.$$

And then by Ht-frame again, there is

$$S \vdash \{\mathsf{True} * Q[u_2/v]\}\, \pi_1(u_2, u)\, \{v.v = u_2 * Q[u_2/v]\}.$$

Apparently, (2b) is derivable using Ht-csq.

2. Now we show that rule

$$\frac{S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P\}\, \pi_1(e, e_1)\, \{v.Q\}}$$

is invalid by a counterexample.

Let $P \equiv \ell \hookrightarrow 0$, $e \equiv \ell \leftarrow 0$, $e_1 \equiv \ell \leftarrow 1$, and $Q \equiv v = () \wedge \ell \hookrightarrow 0$.

It is obvious that $\{P\}\, e\, \{v.Q\}$ holds but $\{P\}\, \pi_1(e, e_1)\, \{v.Q\}$ not, since

$$\{\ell \hookrightarrow 0\}\, \pi_1(\ell \leftarrow 0, \ell \leftarrow 1)\, \{v.v = () \wedge \ell \hookrightarrow 1\}.$$

3. Finally, we add some restrictions on e_1 to let

$$\frac{S \vdash \{P\}\, e\, \{v.Q\} \qquad S \vdash \{P_1\}\, e_1\, \{v.P_2\}}{S \vdash \{P\}\, \pi_1(e, e_1)\, \{v.Q\}}$$

hold.

The idea is that, the expression (e, e_1) is evaluated sequentially, first e then e_1, and $\pi_1(e, e_1)$ selects the evaluation result of e, so the evaluation result of e_1 doesn't matter, and it can refer to the resource claimed by Q but it must not mutate it.

Such P_1 and P_2 can be $P_1 \equiv \exists v.Q$ and $P_2 \equiv Q$.

First, apply Ht-bind to the goal $S \vdash \{P\}\, \pi_1(e, e_1)\, \{v.Q\}$,

$$S \vdash \{Q[u_2/v]\}\, (u_2, e_1)\, \{v_1.Q_1\}, \tag{5}$$
$$S \vdash \{Q_1[u_1/v_1]\}\, \pi_1 u_1\, \{v.Q\}. \tag{6}$$

Then apply Ht-bind to (5),

$$S \vdash \{Q[u_2/v]\}\, e_1\, \{w_1.R_1\}, \tag{7}$$
$$S \vdash \{R_1[w_2/w_1]\}\, (u_2, w_2)\, \{v_1.Q_1\}. \tag{8}$$

Applying Ht-exist and $\forall$-E to the second hypothesis, there is

$$S \vdash \{Q\}\, e_1\, \{v.Q\}.$$

Let

$$\begin{aligned} Q_1 \equiv & v = u_2 * v_1 = (u_2, w_2) * Q[u_2/v], \\ R_1 \equiv & w_1 = u_2 * Q[u_2/v], \end{aligned}$$

then (6), (7) and (8) become

$$S \vdash \{v = u_2 * u_1 = (u_2, w_2) * Q[u_2/v]\}\, \pi_1 u_1\, \{v.Q\}, \tag{6a}$$
$$S \vdash \{Q[u_2/v]\}\, e_1\, \{w_1.w_1 = u_2 * Q[u_2/v]\}, \tag{7a}$$
$$\begin{aligned} S \vdash & \{w_2 = u_2 * Q[u_2/v]\}\, (u_2, w_2) \\ & \quad \{v_1.v = u_2 * v_1 = (u_2, w_2) * Q[u_2/v]\}. \end{aligned} \tag{8a}$$

(7a) is immediate from the second hypothesis.

Substitute (6a) with $u_1 = (u_2, w_2)$ and $v = u_2$, and substitute (8a) with $w_2 = u_2$,

$$S \vdash \{Q\}\, \pi_1(v, w_2)\, \{v.Q\}, \tag{6b}$$
$$S \vdash \{Q[u_2/v]\}\, (u_2, u_2)\, \{v_1.v = u_2 * v_1 = (u_2, u_2) * Q[u_2/v]\}. \tag{8b}$$

(6b) is immediate by Ht-proj, and (8b) is immediate by Ht-ret and Ht-frame.

□

Exercise 4.9. From Ht-If we can derive two, perhaps more natural, rules, which are simpler to use. They require us to only prove a specification of the branch which will be taken.

$$\textsc{Ht-If-True}\quad \frac{\{P * v = \mathtt{true}\}\, e_2\, \{u.Q\}}{\{P * v = \mathtt{true}\}\, \mathtt{if}\ v\ \mathtt{then}\ e_2\ \mathtt{else}\ e_3\, \{u.Q\}}$$

$$\textsc{Ht-If-False}\quad \frac{\{P * v = \mathtt{false}\}\, e_3\, \{u.Q\}}{\{P * v = \mathtt{false}\}\, \mathtt{if}\ v\ \mathtt{then}\ e_2\ \mathtt{else}\ e_3\, \{u.Q\}}$$

Derive Ht-If-True and Ht-If-False from Ht-If.

Proof.

$$\dfrac{\dfrac{\{P * v = \mathtt{true}\}\, e_2\, \{u.Q\}}{\{P * v = \mathtt{true} * v = \mathtt{true}\}\, e_2\, \{u.Q\}} \qquad \dfrac{\dfrac{}{\{\mathsf{False}\}\, e_3\, \{u.Q\}}\,(\textsc{Ht-false})}{\{P * v = \mathtt{true} * v = \mathtt{false}\}\, e_3\, \{u.Q\}}}{\{P * v = \mathtt{true}\}\, \mathtt{if}\ v\ \mathtt{then}\ e_2\ \mathtt{else}\ e_3\, \{u.Q\}}\,(\textsc{Ht-If})$$

$$\dfrac{\dfrac{\dfrac{}{\{\mathsf{False}\}\, e_2\, \{u.Q\}}\,(\textsc{Ht-false})}{\{P * v = \mathtt{false} * v = \mathtt{true}\}\, e_2\, \{u.Q\}} \qquad \dfrac{\{P * v = \mathtt{false}\}\, e_3\, \{u.Q\}}{\{P * v = \mathtt{false} * v = \mathtt{false}\}\, e_3\, \{u.Q\}}}{\{P * v = \mathtt{false}\}\, \mathtt{if}\ v\ \mathtt{then}\ e_2\ \mathtt{else}\ e_3\, \{u.Q\}}\,(\textsc{Ht-If})$$

□

Exercise 4.10. Show the following derived rule for any expression e and $i \in \{1, 2\}$.

$$\frac{S \vdash \{P\}\, e\, \{v.v = \mathtt{inj}_i u * Q\} \qquad S \vdash \{Q[\mathtt{inj}_i u/v]\}\, e_i[u/x_i]\, \{v.R\}}{S \vdash \{P\}\, \mathtt{match}\, e\, \mathtt{with}\, \mathtt{inj}_1 x_1 \Rightarrow e_1 \mid \mathtt{inj}_2 x_2 \Rightarrow e_2\ \mathtt{end}\, \{v.R\}}$$

Proof. It can be derived by Ht-bind and Ht-Match. □

4.1 Derived rules for Hoare triples

Exercise 4.11. Show the following derived rule

$$\textsc{Ht-Pre-Eq}\quad \frac{\Gamma \mid S \vdash \{P[v/x]\}\, e[v/x]\, \{u.Q[v/x]\}}{\Gamma, x : \mathit{Val} \mid S \vdash \{x = v \wedge P\}\, e\, \{u.Q\}}$$

Proof.

$$\dfrac{\dfrac{\dfrac{\Gamma \mid S \vdash \{P[v/x]\}\, e[v/x]\, \{u.Q[v/x]\}}{\Gamma, x : \mathit{Val} \mid S \wedge x = v \vdash \{P[v/x]\}\, e[v/x]\, \{u.Q[v/x]\}} \qquad S \wedge x = v \vdash x = v}{\Gamma, x : \mathit{Val} \mid S \wedge x = v \vdash \{P\}\, e\, \{u.Q\}}\,(\textsc{Eq})}{\Gamma, x : \mathit{Val} \mid S \vdash \{x = v \wedge P\}\, e\, \{u.Q\}}\,(\textsc{Ht-Eq})$$

□

Exercise 4.12. Recall we define the let expression $\texttt{let}\, x = e1\, \texttt{in}\, e2$ using abstraction and application. Show the following derived rule

$$\frac{S \vdash \{P\}\, e_1\, \{x.Q\} \qquad S \vdash \forall v.\{Q[v/x]\}\, e_2[v/x]\, \{u.R\}}{S \vdash \{P\}\, \texttt{let}\, x = e_1\, \texttt{in}\, e_2\, \{u.R\}}\ \textsc{Ht-let}$$

Using this rule is perhaps a bit inconvenient since most of the time the result of evaluating e_1 will be a single value and the postcondition Q will be of the form $x = v \wedge Q'$ for some value v.

The following rule, a special case of the above rule reflects this common case. Derive the rule from the rule Ht-let.

$$\frac{S \vdash \{P\}\, e_1\, \{x.x = v \wedge Q\} \qquad S \vdash \{Q[v/x]\}\, e_2[v/x]\, \{u.R\}}{S \vdash \{P\}\, \texttt{let}\, x = e_1\, \texttt{in}\, e_2\, \{u.R\}}\ \textsc{Ht-let-det}$$

Define the sequencing expression $e_1; e_2$ such that when this expression is run first e_1 is evaluated to a value, the value is discarded, and then e_2 is evaluated. Show the following specifications for the defined construct.

$$\frac{S \vdash \{P\}\, e_1\, \{v.Q\} \qquad S \vdash \{\exists x.Q\}\, e_2\, \{u.R\}}{S \vdash \{P\}\, e_1; e_2\, \{u.R\}}\ \textsc{Ht-seq}$$

$$\frac{S \vdash \{P\}\, e_1\, \{\_.Q\} \qquad S \vdash \{Q\}\, e_2\, \{u.R\}}{S \vdash \{P\}\, e_1; e_2\, \{u.R\}}$$

where $\_.Q$ means that Q does not mention the return value.

Proof. 1. Notice that $\texttt{let}\, x = e_1\, \texttt{in}\, e_2$ is desugared to $(\texttt{rec}\, f(x) = e_2)e_1$, then Ht-let becomes

$$\frac{S \vdash \{P\}\, e_1\, \{x.Q\} \qquad S \vdash \forall v.\{Q[v/x]\}\, e_2[v/x]\, \{u.R\}}{S \vdash \{P\}\, (\texttt{rec}\, f(x) = e_2)e_1\, \{u.R\}}$$

Applying Ht-bind to $S \vdash \{P\}\, (\texttt{rec}\, f(x) = e_2)e_1\, \{u.R\}$, we have to show

$$S \vdash \{P\}\, e_1\, \{x.Q\}, \tag{1}$$
$$S \vdash \forall v.\{Q[x/v]\}\, (\texttt{rec}\, f(x) = e_2)v\, \{u.R\}, \tag{2}$$

where the original x (2) has been substituted by v in order not to be confused with the x in $f(x)$.

(1) is immediate from the first hypothsis. Applying Ht-rec to (2), we have to show

$$\Gamma, g : \mathit{Val} \mid S \wedge \forall v.\{Q\}\, g\, v\, \{u.R\} \vdash \forall v.\{Q[v/x]\}\, e_2[g/f][v/x]\, \{u.R\}$$

Since $\texttt{rec}\, f(x) = e_2$ is not recursive, e_2 does not contain f, we can omit the g, then

$$\Gamma, g : \mathit{Val} \mid S \wedge \forall v.\{Q\}\, g\, v\, \{u.R\} \vdash \forall v.\{Q[v/x]\}\, e_2[v/x]\, \{u.R\}$$

which is immediate from the second hypothsis.

2. Notice that $e_1; e_2$ is desugared to $\texttt{let}\, \_ = e_1\, \texttt{in}\, e_2$, then Ht-seq is immediate from Ht-let-det.

□

Exercise 4.13. Recall we defined $\lambda x.\, e$ to be $\texttt{rec}\, f(x) = e$ where f is some fresh variable. Show the following derived rule.

$$\textsc{Ht-beta}\ \frac{S \vdash \{P\}\, e[v/x]\, \{u.Q\}}{S \vdash \{P\}\, (\lambda x.\, e)v\, \{u.Q\}}$$

Proof. Desugar $(\lambda x.\, e)v$ to $(\texttt{rec}\, f(x) = e)v$, and apply Ht-rec to it, then we have to show

$$\Gamma, g : \mathit{Val} \mid S \wedge \forall v.\{P\}\, g\, v\, \{u.Q\} \vdash \{P\}\, e[v/x]\, \{u.Q\}$$

which is immediate from the hypothesis. □

Exercise 4.14. Derive the following rule.

$$\textsc{Ht-bind-det}\ \frac{E \text{is an eval. context} \qquad S \vdash \{P\}\, e\, \{x.x = u \wedge Q\} \qquad S \vdash \{Q[u/x]\}\, E[u]\, \{w.R\}}{S \vdash \{P\}\, E[e]\, \{w.R\}}$$

Proof. Apply Ht-bind with $Q \equiv x = u \wedge Q$ to the goal, then we have to show

$$S \vdash \forall v.\{u = v \wedge Q[v/x]\}\, E[v]\, \{w.R\}$$

Extract $u = v$ with Ht-eq and substitute v with u in the above entailment,

$$S \wedge u = u \vdash \{Q[u/x]\}\, E[u]\, \{w.R\}$$

which is immediate from the hypothesis. □

Exercise 4.16. Show the following triples and entailments in detail.

1. $$\{R * \ell \hookrightarrow m\}\, \ell \leftarrow !\,\ell + 5\, \{v.R * v = () * \ell \hookrightarrow (m + 5)\}$$

2. $$\{P\}\, e[m + 5/x]\, \{v.Q\} \vdash \{P * \ell \hookrightarrow m\}\, \texttt{let}\, x = !\,\ell + 5\, \texttt{in}\, e\, \{v.Q * \ell \hookrightarrow m\}$$

3. Assuming u does not appear in P and Q show the following entailment.

$$\{P\}\, e[v_1/x]\, \{v.Q\} \vdash \{u = (v_1, v_2) * P\}\, \texttt{let}\, x = \pi_1 u\, \texttt{in}\, e\, \{v.Q\}$$

Proof. 1. Apply Ht-bind-det to the goal

$$\{\ell \hookrightarrow m\}\, \ell \leftarrow !\,\ell + 5\, \{v.v = () * \ell \hookrightarrow (m+5)\} \tag{1}$$

Apply Ht-bind-det to (1),

$$\{\ell \hookrightarrow m\}\; !\,\ell + 5\, \{v.v = v_1 * Q_1\}, \tag{2}$$
$$\{Q_1[v_1/v]\}\, \ell \leftarrow v_1\, \{v.v = () * \ell \hookrightarrow (m+5)\} \tag{3}$$

Apply Ht-bind-det to (2),

$$\{\ell \hookrightarrow m\}\; !\,\ell\, \{v.v = v_2 * Q_2\}, \tag{2a}$$
$$\{Q_2[v_2/v]\}\, v_2 + 5\, \{v.v = v_1 * Q_1\}, \tag{2b}$$

Then by Ht-load, (2a) holds for $v_2 = m$ and $Q_2 \equiv \ell \hookrightarrow m$, and (2b) becomes,

$$\{\ell \hookrightarrow m\}\, m + 5\, \{v.v = v_1 * Q_1\}.$$

By Ht-op, it holds for $v_1 = m + 5$ and $Q_1 \equiv \ell \hookrightarrow m$. Then (3) becomes

$$\{\ell \hookrightarrow m\}\, \ell \leftarrow m + 5\, \{v.v = () * \ell \hookrightarrow (m+5)\}$$

which is obvious by Ht-store.

2. Apply Ht-let-det to the goal

$$\{P\}\, e[m+5/x]\, \{v.Q\} \vdash \{P * \ell \hookrightarrow m\}\; !\,\ell + 5\, \{x.x = v \wedge R\} \tag{4}$$
$$\{P\}\, e[m+5/x]\, \{v.Q\} \vdash \{R[v/x]\}\, e[v/x]\, \{v.Q\} \tag{5}$$

Similar to 1, (4) is immediate for $v = m + 5$ and $R \equiv \ell \hookrightarrow m * P$. Then (5) becomes

$$\{P\}\, e[m+5/x]\, \{v.Q\} \vdash \{\ell \hookrightarrow m * P\}\, e[m+5/x]\, \{v.Q\}$$

which is immediate by apply $*$-weak and Asm.

3. Apply Ht-let-det to the goal,

$$\{P\}\, e[v_1/x]\, \{v.Q\} \vdash \{u = (v_1, v_2) * P\}\, \pi_1 u\, \{x.x = x_1 \wedge R\} \tag{6}$$
$$\{P\}\, e[v_1/x]\, \{v.Q\} \vdash \{R[x_1/x]\}\, e[x_1/x]\, \{v.Q\} \tag{7}$$

Substitute u in (6),

$$\{P\}\, \pi_1(v_1, v_2)\, \{x.x = x_1 \wedge R\}.$$

It is obvious by Ht-proj for $x_1 = v_1$ and $R \equiv P$. Then (7) becomes

$$\{P\}\, e[v_1/x]\, \{v.Q\} \vdash \{P\}\, e[v_1/x]\, \{v.Q\}$$

which is immediate by Asm.

□

4.2 Reasoning about Mutable Data Structures

Exercise 4.19. Explain why the separating conjunction ensures lists are acyclic. What goes wrong if we used ordinary conjunction?

$$\begin{aligned}\mathsf{isList}\, l\, [] &\equiv l = \mathtt{inj}_1\, () \\ \mathsf{isList}\, l\, (x :: xs) &\equiv \exists hd, l'. l = \mathtt{inj}_2\, (hd) * hd \hookrightarrow (x, l') * \mathsf{isList}\, l'\, xs\end{aligned}$$

Proof. By induction on xs, we show that the list l is acyclic, for each node in the list, i.e. except its previous node (if present), there are no other nodes point to it.

For the first case that $xs = []$, it is trivally acyclic.

Then the second case where $xs = x :: xs'$ is interesting. We must show l is acyclic with the induction hypothesis that the all but first node as a sublist l' of l is acyclic.

The "isList" predicate is conjected by 3 separated propositions $l = \mathtt{inj}_2(hd)$, $hd \hookrightarrow (x, l')$ and $\mathsf{isList}\, l'\, xs'$.

The first proposition tells that the list is not empty, the second proposition tells that it points to a head element x and the remainder of the list l', and the third proposition tells that the remainder of the list is also a list.

From the second propsition we know that there cannot be another node in the list points to it because of the points-to predicate $\hookrightarrow$ is exclusive.

From the third propsition and by the induciton hypothesis, we know that l' is acyclic. Put them together, we are confident to claim that the whole list l is acyclic.

If we use ordinary conjunctions, we cannot guarantee that for any node in the list, there are no nodes other than its previous node point to it, so it might be cyclic. $\square$

Exercise 4.20. Let inc denote the following program that increments all values in a linked list of integers:

$$\begin{aligned}\mathtt{rec}\, \mathsf{inc}(l) = {} & \mathtt{match}\, l\, \mathtt{with} \\ & \quad \mathtt{inj}_1\, x_1 \Rightarrow () \\ & \mid \mathtt{inj}_2\, x_2 \Rightarrow \mathtt{let}\, h = \pi_1 \,!\, x_2\, \mathtt{in} \\ & \qquad\qquad\quad \mathtt{let}\, t = \pi_2 \,!\, x_2\, \mathtt{in} \\ & \qquad\qquad\quad x_2 \leftarrow (h + 1, t); \\ & \qquad\qquad\quad \mathsf{inc}\, t \\ & \mathtt{end}\end{aligned}$$

Prove this Hoare triple in detail using the rules just mentioned

$$\forall xs. \forall l. \{\mathsf{isList}\, l\, xs\}\, \mathsf{inc}\, l\, \{v. v = () \wedge \mathsf{isList}\, l\, (\mathsf{map}\, (1+)\, xs)\}$$

where map is defined by

$$\begin{aligned}\mathsf{map}\, f &\equiv [] \\ \mathsf{map}\, (x :: xs) &\equiv f x :: \mathsf{map}\, f\, xs\end{aligned}$$

Proof. Apply HT-REC,

$$\forall xs.\forall l.\{\mathsf{isList}\,l\,xs\}\,f\,l\,\{v.v = () \wedge \mathsf{isList}\,l\,(\mathsf{map}\,(1+)\,xs)\} \vdash$$
$$\forall xs.\forall l.\{\mathsf{isList}\,l\,xs\}\,\texttt{match}\,l\,\texttt{with}\,\ldots\,\texttt{end}\,\{v.v = () \wedge \mathsf{isList}\,l\,(\mathsf{map}\,(1+)\,xs)\} \quad (1)$$

where the body of $\texttt{match}\,l\,\texttt{with}\,\ldots\,\texttt{end}$ has been substituted by f.

There are two cases of xs, let's talk about them one by one.

1. $xs = []$, then $\mathsf{isList}\,l\,[] \equiv l = \texttt{inj}_1\,()$, and $\mathsf{isList}\,l\,(\mathsf{map}\,(1+)\,xs) \equiv \mathsf{isList}\,l\,[] \equiv l = \texttt{inj}_1\,()$. We have to show

$$\forall l.\{l = \texttt{inj}_1\,()\}\,\texttt{match}\,l\,\texttt{with}\,\ldots\,\texttt{end}\,\{v.v = () \wedge l = \texttt{inj}_1\,()\} \qquad (1a)$$

Then apply HT-EQ and EQ to substitute l,

$$\{\mathsf{True}\}\,\texttt{match}\,\texttt{inj}_1\,()\,\texttt{with}\,\texttt{inj}_1\,x_1 \Rightarrow () \mid \texttt{inj}_2\,x_2 \Rightarrow \ldots\,\texttt{end}\,\{v.v = ()\}$$

Apply HT-MATCH,

$$\{\mathsf{True}\}\,()\,\{v.v = ()\}$$

which is immediate from HT-RET.

2. $xs = x :: xs'$, then

$$\begin{aligned} \mathsf{isList}\,l\,x :: xs' &\equiv \exists hd, l'.l = \texttt{inj}_2\,(hd) * hd \hookrightarrow (x, l') * \mathsf{isList}\,l'\,xs', \\ \mathsf{isList}\,l\,(\mathsf{map}\,(1+)\,xs) &\equiv \mathsf{isList}\,l\,(1+x) :: \mathsf{map}\,(1+)\,xs' \\ &\equiv \exists hd, l'.l = \texttt{inj}_2\,(hd) * hd \hookrightarrow (1+x, l') * \mathsf{isList}\,l'\,\mathsf{map}\,(1+)\,xs'. \end{aligned}$$

We have to show

$$\begin{aligned} &\forall xs.\forall l.\{\mathsf{isList}\,l\,xs\}\,f\,l\,\{v.v = () \wedge \mathsf{isList}\,l\,(\mathsf{map}\,(1+)\,xs)\} \vdash \\ &\quad \forall l.\{\exists hd, l'.l = \texttt{inj}_2\,(hd) * hd \hookrightarrow (x, l') * \mathsf{isList}\,l'\,xs'\} \qquad (1b) \\ &\qquad \texttt{match}\,l\,\texttt{with}\,\ldots\,\texttt{end} \\ &\quad \left\{ v.v = () \wedge \exists hd, l'. \begin{pmatrix} l = \texttt{inj}_2\,(hd) * \\ hd \hookrightarrow (1+x, l') * \\ \mathsf{isList}\,l'\,\mathsf{map}\,(1+)\,xs' \end{pmatrix} \right\} \end{aligned}$$

Then apply HT-EXIST, HT-EQ and EQ to substitute l,

$$\begin{aligned} &\forall xs.\forall l.\{\mathsf{isList}\,l\,xs\}\,f\,l\,\{v.v = () \wedge \mathsf{isList}\,l\,(\mathsf{map}\,(1+)\,xs)\} \vdash \\ &\forall hd, l'.\{hd \hookrightarrow (x, l') * \mathsf{isList}\,l'\,xs'\} \\ &\qquad \texttt{match}\,\texttt{inj}_2\,(hd)\,\texttt{with}\,\texttt{inj}_1\,x_1 \Rightarrow () \mid \texttt{inj}_2\,x_2 \Rightarrow \ldots\,\texttt{end} \\ &\quad \{v.v = () \wedge hd \hookrightarrow (1+x, l') * \mathsf{isList}\,l'\,\mathsf{map}\,(1+)\,xs'\} \end{aligned}$$

Apply Ht-match,

$$\begin{aligned}
&\forall xs.\forall l.\{\mathsf{isList}\, l\, xs\}\, f\, l\, \{v.v = () \wedge \mathsf{isList}\, l\, (\mathsf{map}\,(1+)\, xs)\} \vdash \\
&\forall hd, l'.\{hd \hookrightarrow (x, l') * \mathsf{isList}\, l'\, xs'\} \\
&\qquad\qquad \mathtt{let}\, h = \pi_1\, !\, hd\, \mathtt{in} \\
&\qquad\qquad \mathtt{let}\, t = \pi_2\, !\, hd\, \mathtt{in} \\
&\qquad\qquad x_2 \leftarrow (h+1, t); \\
&\qquad\qquad f\, t \\
&\qquad \{v.v = () \wedge hd \hookrightarrow (1+x, l') * \mathsf{isList}\, l'\, \mathsf{map}\,(1+)\, xs'\}
\end{aligned}$$

Apply Ht-let-det twice,

$$\begin{aligned}
&\forall xs.\forall l.\{\mathsf{isList}\, l\, xs\}\, f\, l\, \{v.v = () \wedge \mathsf{isList}\, l\, (\mathsf{map}\,(1+)\, xs)\} \vdash \\
&\forall hd, l'.\{hd \hookrightarrow (x, l') * \mathsf{isList}\, l'\, xs'\} \\
&\qquad\qquad x_2 \leftarrow (x+1, l'); f\, l' \\
&\qquad \{v.v = () \wedge hd \hookrightarrow (1+x, l') * \mathsf{isList}\, l'\, \mathsf{map}\,(1+)\, xs'\}
\end{aligned}$$

With no doubt, it can be derived by applying Ht-load and Ht-store.

□

Exercise 4.23. Derive the rule Ht-Rec-multi.

Ht-Rec-multi

$$\frac{\Gamma, g : \mathit{Val} \mid S \wedge \forall z.\forall v_1.\forall v_2.\{P\}\, g(v_1, v_2)\, \{u.Q\} \vdash \forall z.\forall v_1.\forall v_2.\{P\}\, e[g/f][v_1/x][v_2/y]\, \{u.Q\}}{\Gamma \mid S \vdash \forall z.\forall v.\{\exists v_1, v_2.v = (v_1, v_2) \wedge P\}\, (\mathtt{rec}\, f(x, y) = e)v\, \{u.\exists v_1, v_2.v = (v_1, v_2) \wedge Q\}}$$

where $\mathtt{rec}\, f(x, y) = e$ is the syntax sugar of

$$\mathtt{rec}\, f(p) = \mathtt{let}\, x = \pi_1\, p\, \mathtt{in}\, \mathtt{let}\, y = \pi_2\ \mathtt{in}\, e.$$

and we assume that the variable v is fresh for P and Q

Proof. Apply Ht-exist, Ht-rec and ∀I,

$$\begin{aligned}
&\Gamma, g, v : \mathit{Val} \mid S \wedge \forall z.\forall v_1.\forall v_2\{v = (v_1, v_2) \wedge P\}\, g\, v\, \{u.\exists v_1, v_2.v = (v_1, v_2) \wedge Q\} \vdash \\
&\qquad\qquad \forall z.\forall v_1.\forall v_2\{v = (v_1, v_2) \wedge P\} \\
&\qquad\qquad\qquad\qquad \mathtt{let}\, x = \pi_1\, v\, \mathtt{in}\, \mathtt{let}\, y = \pi_2\, v\, \mathtt{in}\, e[g/f] \\
&\qquad\qquad\qquad \{u.\exists v_1, v_2.v = (v_1, v_2) \wedge Q\}
\end{aligned}$$

Apply Ht-Ht, Ht-eq-pre, and Ht-Ht again, then omit the trivial $\exists v_1', v_2'.(v_1, v_2) = (v_1', v_2')$,

$$\begin{aligned}
&\Gamma, g : \mathit{Val} \mid S \wedge \forall z.\forall v_1.\forall v_2\{P\}\, g\,(v_1, v_2)\, \{u.Q\} \vdash \\
&\qquad\qquad \forall z.\forall v_1.\forall v_2\{P\}\, \mathtt{let}\, x = \pi_1\,(v_1, v_2)\, \mathtt{in}\, \mathtt{let}\, y = \pi_2\,(v_1, v_2)\, \mathtt{in}\, e[g/f]\, \{u.Q\}
\end{aligned}$$

From now on, we omit the context and asumption of the above entailment, since they are the same as those in the hypothesis.

Apply Ht-let-det, Ht-proj and Ht-frame,

$$\forall z.\forall v_1.\forall v_2\{P\}\ \mathtt{let}\ y = \pi_2\,(v_1, v_2)\ \mathtt{in}\ e[g/f][v_1/x]\ \{u.Q\}$$

Again,

$$\forall z.\forall v_1.\forall v_2\{P\}\ e[g/f][v_1/x][v_2/y]\ \{u.Q\}$$

which is immediate from the hypothsis. □

Exercise 4.24. Let append be the following function, which takes two linked lists as arguments and returns a list which is the concatenation of the two.

$$\begin{aligned}\mathtt{rec}\,\mathsf{append}(l, l') = {} & \mathtt{match}\,l\,\mathtt{with} \\ & \quad \mathtt{inj}_1\,x_1 \Rightarrow l' \\ & \mid \mathtt{inj}_2\,x_2 \Rightarrow \mathtt{let}\,p = !\,x_2\,\mathtt{in} \\ & \qquad\qquad\quad \mathtt{let}\,r = \mathsf{append}\,(\pi_2\,p)\,l'\,\mathtt{in} \\ & \qquad\qquad\quad x_2 \leftarrow (\pi_1\,p, r); \\ & \qquad\qquad\quad \mathtt{inj}_2\,x_2 \\ & \mathtt{end}\end{aligned}$$

We wish to give it the following specification where ++ is append on mathematical sequences.

$$\forall xs, ys, l, l'.\{\mathsf{isList}\,l\,xs * \mathsf{isList}\,l' ys\}\ \mathsf{append}\,l\,l'\ \{v.\mathsf{isList}\,v\,(xs +\!\!+\, ys)\}.$$

- Prove the specification.
- Is the following specification also valid?

  $$\forall xs, ys, l, l'.\{\mathsf{isList}\,l\,xs \wedge \mathsf{isList}\,l'\,ys\}\ \mathsf{append}\,l\,l'\ \{v.\mathsf{isList}\,v\,(xs +\!\!+\, ys)\}$$

  Hint: Think about what is the result of append $l\,l'$.

Proof.
- It is trivial to derive this rule from Ht-rec-multi,

  Ht-Rec-multi′

  $$\frac{\Gamma, g : \mathit{Val} \mid S \wedge \forall z.\forall v_1.\forall v_2.\{P\}\ g(v_1, v_2)\ \{u.Q\} \vdash \forall z.\forall v_1.\forall v_2.\{P\}\ e[g/f][v_1/x][v_2/y]\ \{u.Q\}}{\Gamma \mid S \vdash \forall z.\forall v_1.\forall v_2.\{P\}\ (\mathtt{rec}\,f(x, y) = e)\,v_1\,v_2\ \{u.Q\}}$$

  Then apply it to the goal,

  $$\begin{aligned}\Gamma, g : \mathit{Val} \mid {} & \forall xs, ys, l, l'.\{\mathsf{isList}\,l\,xs * \mathsf{isList}\,l' ys\}\ g\,l\,l'\ \{v.\mathsf{isList}\,v\,(xs +\!\!+\, ys)\} \\ & \vdash \forall xs, ys, l, l'.\{\mathsf{isList}\,l\,xs * \mathsf{isList}\,l' ys\} \\ & \qquad\qquad \mathtt{match}\,l\,\mathtt{with} \\ & \qquad\qquad\quad \mathtt{inj}_1\,x_1 \Rightarrow l' \\ & \qquad\qquad \mid \mathtt{inj}_2\,x_2 \Rightarrow \mathtt{let}\,p = !\,x_2\,\mathtt{in} \\ & \qquad\qquad\qquad\qquad\quad \mathtt{let}\,r = g\,(\pi_2\,p)\,l'\,\mathtt{in} \\ & \qquad\qquad\qquad\qquad\quad x_2 \leftarrow (\pi_1\,p, r); \\ & \qquad\qquad\qquad\qquad\quad \mathtt{inj}_2\,x_2 \\ & \qquad\qquad \mathtt{end} \\ & \qquad \{v.\mathsf{isList}\,v\,(xs +\!\!+\, ys)\}\end{aligned}$$

We omit the assumption to reduce verboseness, and consider the two cases of xs in isList.

$$\forall ys, l, l'.\{l = \mathtt{inj}_1\,() * \mathsf{isList}\,l'ys\}\,\mathtt{match}\,l\,\mathtt{with}\,\ldots\,\mathtt{end}\,\{v.\mathsf{isList}\,v\,ys\} \tag{1}$$

$$\begin{gathered}\forall x, xs', hd, l_1, ys, l, l'.\{l = \mathtt{inj}_2\,(hd) * hd \hookrightarrow (x, l_1) * \mathsf{isList}\,l_1\,xs' * \mathsf{isList}\,l'ys\}\\ \mathtt{match}\,l\,\mathtt{with}\,\ldots\,\mathtt{end}\\ \{v.\mathsf{isList}\,v\,(x :: xs' +\!\!+\, ys)\}\end{gathered} \tag{2}$$

Apply $\forall$I and Ht-eq-pre to eliminate l,

$$\forall ys, l'.\{\mathsf{isList}\,l'ys\}\,\mathtt{match}\,\mathtt{inj}_1\,()\,\mathtt{with}\,\ldots\,\mathtt{end}\,\{v.\mathsf{isList}\,v\,ys\} \tag{1a}$$

$$\begin{gathered}\forall x, xs', hd, l_1, ys, l'.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\,l_1\,xs' * \mathsf{isList}\,l'ys\}\\ \mathtt{match}\,\mathtt{inj}_2\,(hd)\,\mathtt{with}\,\ldots\,\mathtt{end}\\ \{v.\mathsf{isList}\,v\,(x :: xs' +\!\!+\, ys)\}\end{gathered} \tag{2a}$$

Then apply Ht-match to extract the arms in `match`,

$$\forall ys, l'.\{\mathsf{isList}\,l'ys\}\;l'\;\{v.\mathsf{isList}\,v\,ys\} \tag{1b}$$

$$\begin{gathered}\forall x, xs', hd, l_1, ys, l'.\;\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\,l_1\,xs' * \mathsf{isList}\,l'ys\}\\ \begin{aligned}&\mathtt{let}\,p = !\,hd\,\mathtt{in}\\ &\mathtt{let}\,r = g\,(\pi_2\,p)\,l'\,\mathtt{in}\\ &hd \leftarrow (\pi_1\,p, r);\\ &\mathtt{inj}_2\,hd\end{aligned}\\ \{v.\mathsf{isList}\,v\,(x :: xs' +\!\!+\, ys)\}\end{gathered} \tag{2b}$$

(1b) is obvious by applying Ht-frame, Ht-ret and the induction hypothesis.

Apply Ht-let-det, Ht-load and Ht-eq-ref on (2b) to eliminate p,

$$\begin{gathered}\forall x, xs', hd, l_1, ys, l'.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\,l_1\,xs' * \mathsf{isList}\,l'ys\}\\ \begin{aligned}&\mathtt{let}\,r = g\,(\pi_2\,(x, l_1))\,l'\,\mathtt{in}\\ &hd \leftarrow (\pi_1\,(x, l_1), r);\\ &\mathtt{inj}_2\,hd\end{aligned}\\ \{v.\mathsf{isList}\,v\,(x :: xs' +\!\!+\, ys)\}\end{gathered}$$

Again, apply Ht-let-det, Ht-bind-det, Ht-proj, Ht-eq-ref to eliminate r,

$$\begin{gathered}\forall x, xs', hd, l_1, ys, l'.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\,l_1\,xs' * \mathsf{isList}\,l'ys\}\\ hd \leftarrow (\pi_1\,(x, l_1), (g\,l_1\,l'));\,\mathtt{inj}_2\,hd\\ \{v.\mathsf{isList}\,v\,(x :: xs' +\!\!+\, ys)\}\end{gathered}$$

Apply Ht-bind-det and Ht-proj to eliminate $\pi_1\,(x, l_1)$, and apply Ht-bind to eliminate $g\,l_1, l'$. Note that we cannot elimitate g immediately because we don't know the value after executing it.

$$\forall x, xs', hd, l_1, ys, l'.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\,l_1\,xs' * \mathsf{isList}\,l' ys\}\ g\,l_1\,l'\ \{u.P\} \tag{2c}$$

$$\forall x, xs', hd, l_1, ys, u.\{P\}\ hd \leftarrow (x, u); \mathtt{inj}_2\,hd\ \{v.\mathsf{isList}\,v\,(x :: xs' +\!\!+ ys)\} \tag{2d}$$

By the induction hypothesis and Ht-frame, we know that (2c) holds for $P \equiv hd \hookrightarrow (x, l_1) * \mathsf{isList}\,u\,(xs' +\!\!+ ys)$. So (2d) becomes

$$\begin{gathered}\forall x, xs', hd, l_1, ys, u.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\,u\,(xs' +\!\!+ ys)\}\\ hd \leftarrow (x, u); \mathtt{inj}_2\,hd\\ \{v.\mathsf{isList}\,v\,(x :: xs' +\!\!+ ys)\}\end{gathered}$$

Apply Ht-store and Ht-Seq,

$$\begin{gathered}\forall x, xs', hd, l_1, ys, v.\{hd \hookrightarrow (x, u) * \mathsf{isList}\,u\,(xs' +\!\!+ ys)\}\\ \mathtt{inj}_2\,hd\\ \{v.v = \mathtt{inj}_2\,hd \wedge \mathsf{isList}\,v\,(x :: xs' +\!\!+ ys)\}\end{gathered}$$

Expand $\mathsf{isList}\,v\,(x :: xs' +\!\!+ ys)$, then it becomes

$$\begin{gathered}\forall x, xs', hd, l_1, ys, v.\{hd \hookrightarrow (x, u) * \mathsf{isList}\,u\,(xs' +\!\!+ ys)\}\\ \mathtt{inj}_2\,hd\\ \left\{v.v = \mathtt{inj}_2\,hd \wedge \exists hd', l_1'.\ \begin{array}{l} v = \mathtt{inj}_2(hd') * \\ hd' \hookrightarrow (x, l_1') * \\ \mathsf{isList}\,l_1'\,(xs' +\!\!+ ys)\end{array}\right\}\end{gathered}$$

Apparently, $\exists hd, u$ such that

$$\begin{gathered}\forall x, xs', hd, l_1, ys, v.\{hd \hookrightarrow (x, u) * \mathsf{isList}\,u\,(xs' +\!\!+ ys)\}\\ \mathtt{inj}_2\,hd\\ \{v.v = \mathtt{inj}_2\,hd * hd \hookrightarrow (x, u) * \mathsf{isList}\,u\,(xs' +\!\!+ ys)\}\end{gathered}$$

which is immediately by Ht-frame and Ht-ret.

- It is not valid. Because the ordinary conjuntion cannot guarantee that list l and l' don't overlap with each other. When there are overlaps, $\mathsf{append}\,l\,l'$ can construct a cyclic list.

□

Exercise 4.25. The append function in the previous exercise is not tail recursive and hence it space consumption is linear in the length of the first list. A better implementation of append for linked lists is the following.

$$\begin{aligned}\mathsf{append}'\,l\,l' = \texttt{let}\,go = \texttt{rec}\,f(h\,p) = {}& \texttt{match}\,h\,\texttt{with}\\ &\quad \texttt{inj}_1\,x_1 \Rightarrow p \leftarrow (\pi_1\,(!\,p), l')\\ &\mid \texttt{inj}_2\,x_2 \Rightarrow f\,(\pi_2\,(!\,x_2))\,x_2\\ &\texttt{end}\\ \texttt{in}\;\texttt{match}\,l\,\texttt{with}&\\ \quad \texttt{inj}_1\,x_1 \Rightarrow l'&\\ \mid \texttt{inj}_2\,x_2 \Rightarrow go\,(\pi_2(!\,x_2))\,x_2;\; l&\\ \texttt{end}&\end{aligned}$$

In the function go the value p is the last node of the list l we have seen while traversing l. Thus go traverses the first list and once it reaches the end it updates the tail pointer in the last node to point to the second list, l' .

Prove for append' the same specification as for append above. You need to come up with a strong enough invariant for the function go, relating h, p and xs and ys.

Proof. We are going to prove

$$\forall xs, ys, l, l'.\{\mathsf{isList}\,l\,xs * \mathsf{isList}\,l' ys\}\;\mathsf{append}'\,l\,l'\;\{v.\mathsf{isList}\,v\,(xs +\!\!+\, ys)\}. \tag{1}$$

First, consider this recursive function

$$\begin{aligned}\texttt{rec}\,f(h\,p) = {}& \texttt{match}\,h\,\texttt{with}\\ &\quad \texttt{inj}_1\,x_1 \Rightarrow p \leftarrow (\pi_1\,(!\,p), l')\\ &\mid \texttt{inj}_2\,x_2 \Rightarrow f\,(\pi_2\,(!\,x_2))\,x_2\\ &\texttt{end}\end{aligned}$$

From the first arm of `match` we know that l' is appended after at p which is a pointer of a node, at the end of the function.

From the second arm of `match` we know that h is the "next" pointer of p, which is optional, and after an iteration, both h and p move to the next node.

Based on these observations, we construct the following invariant of f and prove it.

$$\begin{aligned}\forall l', x, h, p, xs, ys.\{&p \hookrightarrow (x, h) * \mathsf{isList}\,h\,xs * \mathsf{isList}\,l'\,ys\}\\ &f\,h\,p\\ &\{v.v = () \wedge \exists h.p \hookrightarrow (x, h).\mathsf{isList}\,h\,(xs +\!\!+\, ys)\}\end{aligned} \tag{2}$$

Here h might change after execution of $f\,h\,p$, but p always remains unchanged.

Apply Ht-rec, and we have the induction hypothesis and the new goal

$$\begin{aligned}
\Gamma, g : \mathit{Val} \mid \forall l', x, h, p, xs, ys. &\{p \hookrightarrow (x, h) * \mathsf{isList}\, h\, xs * \mathsf{isList}\, l'\, ys\} && (3)\\
&\quad g\, h\, p\\
&\quad\quad \{v. v = () \land \exists h. p \hookrightarrow (x, h) * \mathsf{isList}\, h\, (xs \mathbin{+\!\!+} ys)\}\\
\vdash \forall l', x, h, p, xs, ys. &\{p \hookrightarrow (x, h) * \mathsf{isList}\, h\, xs * \mathsf{isList}\, l'\, ys\} && (4)\\
&\quad \mathtt{match}\, h\, \mathtt{with}\\
&\quad\quad \mathtt{inj}_1\, x_1 \Rightarrow p \leftarrow (\pi_1\,(!\,p), l')\\
&\quad \mid \mathtt{inj}_2\, x_2 \Rightarrow g\,(\pi_2\,(!\,x_2))\, x_2\\
&\quad \mathtt{end}\\
&\quad\quad \{v. v = () \land \exists h. p \hookrightarrow (x, h) * \mathsf{isList}\, h\, (xs \mathbin{+\!\!+} ys)\}
\end{aligned}$$

Consider 2 cases of xs and apply Ht-eq-pre, Ht-match and Ht-exist on (4),

$$\begin{aligned}
&\forall l', x, p, ys. \{p \hookrightarrow (x, \mathtt{inj}_1\,()) * \mathsf{isList}\, l'\, ys\} && (4a)\\
&\quad p \leftarrow (\pi_1\,(!\,p), l')\\
&\quad \{v. v = () \land \exists h. p \hookrightarrow (x, h) * \mathsf{isList}\, h\, ys\}
\end{aligned}$$

$$\begin{aligned}
&\forall l', x, x', hd, h', p, xs', ys. \left\{\begin{array}{l} p \hookrightarrow (x, \mathtt{inj}_2\,(hd)) * \\ hd \hookrightarrow (x', h') * \\ \mathsf{isList}\, h'\, xs' * \\ \mathsf{isList}\, l'\, ys \end{array}\right\}\\
&\quad g\,(\pi_2\,(!\,hd))\, hd\\
&\quad \{v. v = () \land \exists h. p \hookrightarrow (x, h) * \mathsf{isList}\, h\, (x' :: xs' \mathbin{+\!\!+} ys)\} && (4b)
\end{aligned}$$

(4a) can be obtained by applying Ht-bind-det, Ht-load, Ht-bind-det, Ht-proj, and Ht-store, where the last step is

$$\begin{aligned}
&\forall l', x, p, ys. \{p \hookrightarrow (x, \mathtt{inj}_1\,()) * \mathsf{isList}\, l'\, ys\}\\
&\quad p \leftarrow x, l')\\
&\quad \{v. v = () \land p \hookrightarrow (x, l') * \mathsf{isList}\, l'\, ys\}
\end{aligned}$$

which is immediate from Ht-store.

Similarly, apply Ht-bind-det, Ht-load, Ht-bind-det, and Ht-proj to eliminate hd in it,

$$\begin{aligned}
&\forall l', x, x', hd, h', p, xs', ys.\\
&\{p \hookrightarrow (x, \mathtt{inj}_2\,(hd)) * hd \hookrightarrow (x', h') * \mathsf{isList}\, h'\, xs' * \mathsf{isList}\, l'\, ys\}\\
&\quad g\, h'\, hd\\
&\{v. v = () \land \exists h. p \hookrightarrow (x, h) * \mathsf{isList}\, h\, (x' :: xs' \mathbin{+\!\!+} ys)\}
\end{aligned}$$

Let $h \equiv \mathtt{inj}_2\,(hd)$ and $xs \equiv x' :: xs'$, and fold the first isList, then we have

$$\begin{aligned}
&\forall l', x, h, p, xs, ys. \{p \hookrightarrow (x, h) * \mathsf{isList}\, h\, xs * \mathsf{isList}\, l'\, ys\}\\
&\quad g\, h'\, hd\\
&\quad\quad \{v. v = () \land \exists h. p \hookrightarrow (x, h) * \mathsf{isList}\, h\, (x' :: xs' \mathbin{+\!\!+} ys)\}
\end{aligned}$$

It is immediate from the induction hypothesis (3).

Now we can use (2) to prove (1). First apply HT-LET-DET to eliminate *go*,

$$\forall xs, ys, l, l'.\{\mathsf{isList}\, l\, xs * \mathsf{isList}\, l' ys\} \quad (5)$$

$$\begin{array}{l}\texttt{match}\, l\, \texttt{with} \\ \quad \texttt{inj}_1\, x_1 \Rightarrow l' \\ \mid \texttt{inj}_2\, x_2 \Rightarrow (\texttt{rec}\, f(h\, p) = \ldots)\, (\pi_2\, !\, (x_2))\, x_2;\ l \\ \texttt{end}\end{array}$$

$$\{v.\mathsf{isList}\, v\, (xs +\!\!+\, ys)\}$$

Consider 2 cases of xs, and apply HT-EQ-PRE, HT-MATCH, and HT-EXIST on (5)

$$\forall ys, l, l'.\{\mathsf{isList}\, l' ys\}\ l'\ \{v.\mathsf{isList}\, v\, ys\} \quad (5a)$$

$$\forall x, xs', ys, hd, l_1, l'.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\, l_1\, xs' * \mathsf{isList}\, l' ys\} \quad (5b)$$

$$(\texttt{rec}\, f(h\, p) = \ldots)\, (\pi_2\, !\, (hd))\, hd;\ \texttt{inj}_2\, hd$$

$$\{v.\mathsf{isList}\, v\, (x :: xs' +\!\!+\, ys)\}$$

(5a) is immediate by HT-RET.

Apply HT-BIND-DET, HT-LOAD, HT-BIND-DET, and HT-PROJ to eliminate $\pi_2\, !\, (hd)$

$$\forall x, xs', ys, hd, l_1, l'.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\, l_1\, xs' * \mathsf{isList}\, l' ys\} \quad (5c)$$

$$(\texttt{rec}\, f(h\, p) = \ldots)\, l_1\, hd;\ \texttt{inj}_2\, hd$$

$$\{v.\mathsf{isList}\, v\, (x :: xs' +\!\!+\, ys)\}$$

From (2) we know that

$$\forall x, xs', ys, hd, l_1, l'.\{hd \hookrightarrow (x, l_1) * \mathsf{isList}\, l_1\, xs' * \mathsf{isList}\, l' ys\} \quad (5d)$$

$$(\texttt{rec}\, f(h\, p) = \ldots)\, l_1\, hd$$

$$\{v.v = () \land \exists h.hd \hookrightarrow (x, h) * \mathsf{isList}\, h\, (xs' +\!\!+\, ys)\}$$

Then apply HT-SEQ to (5c) and associate it with (5d)

$$\forall x, xs', ys, hd, l_1, l'.\{\exists h.hd \hookrightarrow (x, h) * \mathsf{isList}\, h\, xs' * \mathsf{isList}\, l' ys\}$$

$$\texttt{inj}_2\, hd$$

$$\{v.\mathsf{isList}\, v\, (x :: xs' +\!\!+\, ys)\}$$

It can be derived by HT-EXIST, HT-RET, and the definition of isList. ☐

Exercises 4.26 – 4.30 are done in Coq.

4.3 Abstract Data Types

Exercises of this section are done in Coq.

5 Case Study: foldr

Exercises of this chapter are done in Coq.