Incorrectness Specification Inference

ANONYMOUS AUTHOR(S)

1 MOTIVATION EXAMPLE: CONCAT

For the concat example,

```
let rec concat s1 s2 =
  match s1 with
  | [] -> s2
  | h1 :: t1 ->
    let s3 = concat t1 s2 in
    h1 :: s3
```

we expect that the overapproximate triple $\{\Sigma\} concat\ s_1\ s_2 \downarrow v \{\Phi\}$ hold, where:

$$\Sigma \equiv \neg dup(s_1) \land \neg dup(s_2)$$
$$\Phi \equiv \neg dup(v)$$
$$dup(s) \equiv \text{ the stack } s \text{ contains some elements appearing for 2 or more times.}$$
$$emp(s) \equiv \text{ the stack } s \text{ is empty.}$$

However, this triple is not valid (e.g. $concat\ [1;2]\ [2;3] = [1;2;2;3]$) which means the program is buggy. Thus we expect to find an underapproximate triple as its incorrectness specification that summarize the buggy executions of *concat*. We cannot simply negate the postcondition Φ to build a triple $[\Sigma] concat\ s_1\ s_2 \downarrow v [\neg\Phi]$ because there exists unreachable state in the negated postcondition $\neg\Phi$. For example, the stack $v \equiv [1;1;1]$ is not reachable, because all elements in the output stack should be contained by the two input stacks s_1 and s_2, additionally there are three 1s in the output stack, thus at least one of the input stack will contains two 1s which conflicts with the precondition Σ that asks both input stacks having no duplicate element.

1.1 Problem

Thus our goal is to infer P and Q such that:

(1) $[P] concat\ s_1\ s_2 \downarrow v[Q]$;
(2) $P \implies \Sigma$;
(3) $Q \implies \neg\Phi$.

1.2 Solution

One possible result is:

$$P_0 \equiv \neg dup(s_1) \land \neg dup(s_2) \land \exists u, mem(s_1, u) \land mem(s_2, u)$$
$$Q_0 \equiv dup(v) \land \neg dup3(v)$$

where $dup3(s)$ means the stack s contains some elements appearing for 3 or more times.

2022. XXXX-XXXX/2022/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.3 Proof fails without inductive invariant

Although P_0 and Q_0 is correct but we cannot verify the triple $[P_0] concat\ s_1\ s_2 \downarrow v [Q_0]$ directly, because P_0 and Q_0 is not inductive. More precisely, we expect that,

$$[P] T [ok : Q]$$

where T is an imperative version (but with recursion) of *concat*:

$$\begin{aligned} T :=& \\ & (\text{assume } s_1 = []; v := s_2) \oplus \\ & (\text{assume } \neg(s_1 := []); (h_1, t_1) := cons^{-1}(s_1); s_3 := T(t_1, s_2); v := h_1 :: s_3) \end{aligned}$$

Now we do not consider exceptions, thus we label Q as ok (and omitted if the context is clear) and expect the program can terminate normally but reach state over from Φ. Now we show, if we do not introduce the new inductive invariant, the proof will fail.

Notice that, the proof tree actually starts from the post condition $P(s_1, s_2) \land Q(v)$ instead of $Q(v)$, because the incorrectness logic need constraint all variables in the postcondition. Thus the input arguments should be constrained by the precondition $P(s_1, s_2)$. In summary, we add a new rule:

$$\frac{[P(\overline{x})]\overline{y} = T(\overline{x})[P(\overline{x}) \land Q(\overline{y})]}{Spec(T, (P, Q))}\ \text{Specification}$$

Daully, we add an application rule:

$$\frac{Spec(T, (P, Q))}{\forall \overline{x}\ \overline{y}, [P(\overline{x})]\overline{y} = T(\overline{x})[P(\overline{x}) \land Q(\overline{y})]}\ \text{Apply}$$

Now we have

$$\dfrac{\dfrac{\dfrac{}{\begin{array}{l}[P(s_1, s_2)] \\ \text{assume } \neg(s_1 := []); \\ [P(s_1, s_2) \land s_1 \neq []]\end{array}}\ \text{Assume} \qquad \dfrac{\dfrac{}{\begin{array}{l}[P(s_1, s_2) \land s_1 \neq []] \\ (h_1, t_1) := cons^{-1}(s_1) \\ [P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1]\end{array}}\ \text{Assign} \qquad \begin{array}{l}[P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1] \\ s_3 := T(t_1, s_2); v := h_1 :: s_3 \\ [\exists ... \land Q(v)]\end{array}}{[P(s_1, s_2) \land s_1 \neq []](h_1, t_1) := cons^{-1}(s_1); ...[\exists ... \land Q(v)]}\ \text{Seq}}{\dfrac{[P(s_1, s_2)]\text{assume } \neg(s_1 := []); (h_1, t_1) := cons^{-1}(s_1); ...[\exists ... \land Q(v)]}{\dfrac{[P(s_1, s_2)]T(s_1, s_2) = v[P(s_1, s_2) \land Q(v)]}{Spec(T, (P, Q))}\ \text{Spec}}\ \text{Choice: 1}}\ \text{Seq}$$

We choose the second branch of T, because the first branch will lead $P(s_1, s_2) \land s1 = [] \iff \bot$.

$$\dfrac{\dfrac{\dfrac{Spec(T, (P, Q))}{\begin{array}{l}[P(t_1, s_2)]s_3 := \\ T(t_1, s_2)[Q(s_3)]\end{array}}\ \text{Apply} \qquad \begin{array}{l}P(t_1, s_2) \implies \\ P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1\end{array}}{\begin{array}{l}[P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1]s_3 := \\ T(t_1, s_2) \\ [P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1 \land Q(s_3)]\end{array}}\ \text{Cons} \qquad \dfrac{\dfrac{}{\begin{array}{l}[... \land Q(s_3)] \\ v := h_1 :: s_3 \\ [.. \land Q(s_3) \land v = h_1 :: s_3]\end{array}}\ \text{Assign} \qquad \begin{array}{l}\exists s_3\ h_1\ t_1, P(s_1, s_2) \land Q(v) \implies \\ P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1 \\ \land Q(s_3) \land v = h_1 :: s_3\end{array}}{\begin{array}{l}[P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: \\ t_1 \land Q(s_3)]\ v := h_1 :: s_3 \\ [\exists ... \land Q(v)]\end{array}}\ \text{Con}}{\begin{array}{l}[P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1] \\ s_3 := T(t_1, s_2); v := h_1 :: s_3\ [\exists ... \land Q(v)]\end{array}}\ \text{Seq}$$

When we encounter the recursive calling $T(t_1, s_2)$, to apply the triple $[P(t_1, s_2)]s_3 := T(t_1, s_2)[Q(s_3)]$ to the current state, we use the consequence rule of the incorrectness logic. It requires

$$P(t_1, s_2) \implies P(s_1, s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1$$

The precondition $P(t_1, s_2)$ does not constraint the variable s_1 and h_1 because they are not the arguments of $T(t_1, s_2)$, which make the proof fail. It push us to add the frame rule to the application rule:

$$\frac{Spec(T, (P, Q)), \overline{z} \cap \overline{x} = \emptyset}{\forall \overline{x}\ \overline{y}, [R(\overline{z}) \land P(\overline{x})]\overline{y} = T(\overline{x})[R(\overline{z}) \land P(\overline{x}) \land Q(\overline{y})]}\ \text{Apply-Frame-First-Try}$$

which follows the typical frame rule that asks the variable set $\overline{z}$ is disjoint with the variable set $\overline{x}$. However, even with the new application rule, we cannot prove the implication as the body of the implication includes:

$$s_1 = h_1 :: t_1$$

which is a relation between the arguments of calling function ($\overline{x}$) can the framed variables ($\overline{z}$). To solve this, I introduce a stronger application rule:

$$\frac{Spec(T,(P,Q)), \overline{z} \cap \overline{x} = \emptyset \quad P(\overline{x}) \wedge Q(\overline{y}) \implies \exists \overline{z}, R(\overline{x},\overline{y},\overline{z}) \wedge P(\overline{x}) \wedge Q(\overline{y})}{\forall \overline{x}\ \overline{y}, [R(\overline{x},\overline{z}) \wedge P(\overline{x})]\overline{y} = T(\overline{x})[R(\overline{x},\overline{y},\overline{z}) \wedge P(\overline{x}) \wedge Q(\overline{y})]}\ \text{Apply-Frame}$$

where R can be built from any variables, but we do not allow any state in $P(\overline{x}) \wedge Q(\overline{y})$ to be invalid after framing.

Now we set $R(\overline{x},\overline{y},\overline{z})$ as $s_1 = h_1 :: t_1$ and we just need to prove:

$$\begin{aligned}
&P(t_1,s_2) \wedge s_1 = h_1 :: t_1 \implies P(s_1,s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \\
&[P(t_1,s_2) \wedge s_1 = h_1 :: t_1]s_3 = T(t_1,s_2)[P(s_1,s_2) \wedge Q(t_1,s_2) \wedge s_1 = h_1 :: t_1] \\
&P(s_1,s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \wedge Q(s_3) \implies P(s_1,s_2) \wedge Q(t_1,s_2) \wedge s_1 = h_1 :: t_1
\end{aligned}$$

However, we can still not prove it, as we do not know if $\exists u, mem(s_1,u) \wedge mem(s_2,u)$ (required by $P(s_1,s_2)$). One counter example is

$$\begin{aligned}
&s_1 \equiv [1;1], s_2 \equiv [1;3], \\
&h_1 \equiv 1, t_1 \equiv [1]
\end{aligned}$$

which means the precondition $P(s_1,s_2)$ should be strengthen (or $P(s_1,s_2)$ should be weaken). On the other hand, we encounter another similar implication when we try to finish the proof:

$$P(s_1,s_2) \wedge Q(v) \implies P(s_1,s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \wedge Q(s_3) \wedge v = h_1 :: s_3$$

We first introduce a new specification rule:

$$\frac{P(\overline{x})]\overline{y} = T(\overline{x})[R(\overline{x},\overline{y},\overline{z}) \wedge P(\overline{x}) \wedge Q(\overline{y})] \quad P(\overline{x}) \wedge Q(\overline{y}) \implies \exists \overline{z}, R(\overline{x},\overline{y},\overline{z}) \wedge P(\overline{x}) \wedge Q(\overline{y})}{Spec(T,(P,Q))}\ \text{Spec-Frame}$$

Now we set $R(\overline{x},\overline{y},\overline{z})$ as $s_1 = h_1 :: t_1 \wedge v = h_1 :: s_3$ and we just need to prove:

$$P(s_1,s_2) \wedge Q(v) \wedge s_1 = h_1 :: t_1 \wedge v = h_1 :: s_3 \implies P(s_1,s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \wedge Q(s_3) \wedge v = h_1 :: s_3$$

Still, we cannot prove it, as we do not know if $dup(s_3)$ (required by $Q(s_3)$). One counter example is:

$$\begin{aligned}
&s_1 \equiv [1;2], s_2 \equiv [1;3], v = [1;2;1;3] \\
&h_1 \equiv 1, t_1 \equiv [2], s_3 = [2;1;3]
\end{aligned}$$

which means the postcondition $Q(v)$ should be strengthen (or $Q(s_3)$ should be weaken). Let's look deeper to make the problem clear. This counterexample will happen during the execution (unreachable from *concat* $[1;2]$ $[1;3]$). Thus we expect $Q(s_3)$ to be consistent with $[2;1;3]$:

$$s_3 \equiv [2;1;3] \models Q(s_3) \tag{1}$$

However, we also want $Q(s_3) \implies \neg\Phi$, thus

$$s_3 \equiv [2;1;3] \models dup(s_3) \tag{2}$$

which is impossible, thus (P,Q) is not inductive,

1.4 Cannot find an inductive invariant

Assume there exists an inductive invariant (P_I, Q_I) can help to prove (P, Q). Following the conseqeunce rule, we expect:

$$\frac{P_I \implies P \qquad [P_I]C[Q_I] \qquad Q \implies Q_I}{[P]C[Q]}\ \text{Con}$$

where

$$P_I(s_1, s_2) \wedge Q_I(v) \wedge s_1 = h_1 :: t_1 \wedge v = h_1 :: s_3 \implies P_I(s_1, s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \wedge Q_I(s_3) \wedge v = h_1 :: s_3$$

Notice that, the new postcondition Q_I is not required to be a subset of $\neg\Phi$. There are two reachable paths:

$$\begin{aligned} &s_1 \equiv [1;2], s_2 \equiv [1;3], v = [1;2;1;3] \\ &h_1 \equiv 1, t_1 \equiv [2], s_3 = [2;1;3] \end{aligned} \tag{X_1}$$

and

$$\begin{aligned} &s_1 \equiv [2;1], s_2 \equiv [1;3], v = [2;1;1;3] \\ &h_1 \equiv 2, t_1 \equiv [1], s_3 = [1;1;3] \end{aligned} \tag{X_2}$$

Thus we expect Q_I consistent with these four concrete values: $[2;1;3]$, $[1;2;1;3]$, $[1;1;3]$ and $[2;1;1;3]$. Then Q_I contains both duplicate stacks and non-duplicate stacks, and if a stack s_3 can satisfy Q_I dependents on if h_1 is a member of s_3. If we do not know the value of h_1, we cannot decide if a given stack can satisfy Q_I. However, h_1 is not belong to the arguments or the result of calling $s_3 = T(t_1, s_2)$.

1.5 Mutual inductive invariant

I believe the problem is that, we want Q_I to include both duplicate stacks and non-duplicate stacks. Alternatively, if we have two inductive invariants:

$$\begin{aligned} &P_1 \equiv \neg dup(s_1) \wedge \neg dup(s_2) \wedge \exists u, mem(s_1, u) \wedge mem(s_2, u) \\ &Q_1 \equiv \neg dup3(v) \wedge dup(v) \\ &[P_1] v = T(s_1, s_2) [Q_1] \\ &P_2 \equiv \neg dup(s_1) \wedge \neg dup(s_2) \wedge \forall u, \neg mem(s_1, u) \vee \neg mem(s_2, u) \\ &Q_2 \equiv \neg dup3(v) \wedge \neg dup(v) \\ &[P_2] v = T(s_1, s_2) [Q_2] \end{aligned}$$

These two triples have disjointed precondition (with guard $\exists u, mem(s_1, u) \wedge mem(s_2, u)$) and disjointed postcondition (with guard $dup(v)$).

Now we revisit the proof tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\begin{array}{l}[P_1(s_1,s_2)]\\ \text{assume } \neg(s_1 := []);\\ [P_1(s_1,s_2) \wedge s_1 \neq []]\end{array}}\ \text{Assume} \qquad \dfrac{\dfrac{}{\begin{array}{l}[P_1(s_1,s_2) \wedge s_1 \neq []]\\ (h_1,t_1) := cons^{-1}(s_1)\\ [P_1(s_1,s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1]\end{array}}\ \text{Assign} \qquad \dfrac{\text{To be proved}}{\begin{array}{l}[P_1(s_1,s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1]\\ s_3 := T(t_1,s_2); v := h_1 :: s_3\\ [\ldots \wedge Q_1(v)]\end{array}}}{[P_1(s_1,s_2) \wedge s_1 \neq []](h_1,t_1) := cons^{-1}(s_1); \ldots [\ldots \wedge Q_1(v)]}\ \text{Seq}}{[P_1(s_1,s_2)]\text{assume } \neg(s_1 := []); (h_1,t_1) := cons^{-1}(s_1); \ldots [\ldots \wedge Q_1(v)]}\ \text{Seq}}{[P_1(s_1,s_2)]T(s_1,s_2) = v[F \wedge P_1(s_1,s_2) \wedge Q_1(v)]}\ \text{Choice: 1}}{Spec(T, (P_1, Q_1))}\ \text{Spec-Frame}$$

where F is the framing formula (we will set it later).

In order to distinguish the two concrete executions X_1 and X_2, we use the disjunction rule

$$\dfrac{[P_1]C[Q_1] \qquad [P_2]C[Q_2]}{[P_1 \lor P_2]C[Q_1 \lor Q_2]}\ \text{Disjunction}$$

to split current precondition $P_1(s_1,s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1$ by the guard $\exists u, mem(s_1,u) \land mem(s_2,u)$ to decide to apply which inductive invariant:

$$\dfrac{\begin{array}{l}[P_1(s_1,s_2)... \land \exists u, mem(t_1,u) \land mem(s_2,u)]\\ s_3 := T(t_1,s_2); v := h_1 :: s_3\\ [{\color{red}F} \land P_1(s_1,s_2) \land Q_1(v) \land \exists u, mem(t_1,u) \land mem(s_2,u)]\end{array} \qquad \begin{array}{l}[P_1(s_1,s_2)... \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\\ s_3 := T(t_1,s_2); v := h_1 :: s_3\\ [{\color{red}F} \land P_1(s_1,s_2) \land Q_1(v) \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\end{array}}{\begin{array}{l}[P_1(s_1,s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1]\\ s_3 := T(t_1,s_2); v := h_1 :: s_3\\ [{\color{red}F} \land P_1(s_1,s_2) \land Q_1(v)]\end{array}}\ \text{Disj}$$

where we plan to apply $[P_1]T[Q_1]$ in the first case and $[P_1]T[Q_1]$ in the second case. However, we want to eliminate the unreachable states from the postcondition ${\color{red}F} \land P_1(s_1,s_2) \land Q_1(v)$. For example, in the first case, after we apply the $[P_1]T[Q_1]$, the predicate $dup(s_3)$ should hold. Thus we set ${\color{red}F}$ as:

$$\begin{aligned}{\color{red}F} \equiv\ & s_1 = h_1 :: t_1 \land v = h_1 :: s_3 \land \\ & ((\exists u, mem(t_1,u) \land mem(s_2,u)) \implies (dup(s_3) \land \neg dup3(s_3))) \land \\ & (\neg(\exists u, mem(t_1,u) \land mem(s_2,u)) \implies (\neg dup(s_3) \land \neg dup3(s_3) \land mem(s_2,h_1)))\end{aligned}$$

For the first situation we have:

$$\dfrac{\dfrac{\dfrac{Spec(T,(P_1,Q_1))}{\begin{array}{l}[P_1(t_1,s_2) \land s_1 = h_1 ::\\ t_1 \land \neg mem(t_1,h_1) \land \exists u, mem(t_1,u) \land mem(s_2,u)]\\ s_3 := T(t_1,s_2)\\ [P_1(t_1,s_2) \land Q_1(s_3) \land s_1 = h_1 ::\\ t_1 \land \neg mem(t_1,h_1) \land \exists u, mem(t_1,u) \land mem(s_2,u)]\end{array}}\ \text{Apply-Frame} \qquad {\color{red}\phi_a}}{\begin{array}{l}[... \land \exists u, mem(t_1,u) \land mem(s_2,u)]\\ s_3 := T(t_1,s_2)\\ [... \land \exists u, mem(t_1,u) \land mem(s_2,u) \land Q_1(s_3))]\end{array}}\ \text{Cons} \qquad \dfrac{\dfrac{}{\begin{array}{l}[...]\\ v := h_1 :: s_3\\ [.. \land v = h_1 :: s_3]\end{array}}\ \text{Assign} \qquad {\color{red}\phi_b}}{\begin{array}{l}[... \land \exists u, mem(t_1,u) \land mem(s_2,u) \land Q_1(s_3))]\\ v := h_1 :: s_3\\ [... \land Q_1(v) \land \exists u, mem(t_1,u) \land mem(s_2,u)]\end{array}}\ \text{Con}}{\begin{array}{l}[P_1(s_1,s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1 \land \exists u, mem(t_1,u) \land mem(s_2,u)]\\ s_3 := T(t_1,s_2); v := h_1 :: s_3\\ [{\color{red}F} \land P_1(s_1,s_2) \land Q_1(v) \land \exists u, mem(t_1,u) \land mem(s_2,u)]\end{array}}\ \text{Seq}$$

where:

$$\begin{aligned}&{\color{red}\phi_a} \equiv\\ &\quad P_1(t_1,s_2) \land s_1 = h_1 :: t_1 \land \neg mem(t_1,h_1) \land \exists u, mem(t_1,u) \land mem(s_2,u) \implies\\ &\quad P_1(s_1,s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1 \land \exists u, mem(t_1,u) \land mem(s_2,u)\\ &{\color{red}\phi_b} \equiv\\ &\quad {\color{red}F} \land P_1(s_1,s_2) \land Q_1(v) \land \exists u, mem(t_1,u) \land mem(s_2,u) \implies\\ &\quad P_1(s_1,s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1 \land \exists u, mem(t_1,u) \land mem(s_2,u) \land Q_1(s_3) \land v = h_1 :: s_3\end{aligned}$$

It is easy to check that these two implication are valid.

For the second situation we have:

$$\dfrac{\dfrac{\dfrac{Spec(T,(P_2,Q_2))}{\begin{array}{l}[P_2(t_1,s_2) \land s_1 = h_1 ::\\ t_1 \land \neg mem(t_1,h_1) \land mem(s_2,h_1) \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\\ s_3 := T(t_1,s_2)\\ [P_2(t_1,s_2) \land Q_2(s_3) \land s_1 = h_1 ::\\ t_1 \land \neg mem(t_1,h_1) \land mem(s_2,h_1) \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\end{array}}\ \text{Apply-Frame} \qquad {\color{red}\phi_a}}{\begin{array}{l}[... \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\\ s_3 := T(t_1,s_2)\\ [... \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u)) \land Q_2(s_3))]\end{array}}\ \text{Cons} \qquad \dfrac{\dfrac{}{\begin{array}{l}[...]\\ v := h_1 :: s_3\\ [.. \land v = h_1 :: s_3]\end{array}}\ \text{Assign} \qquad {\color{red}\phi_b}}{\begin{array}{l}[... \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u)) \land Q_1(s_3))]\\ v := h_1 :: s_3\\ [... \land Q_1(v) \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\end{array}}\ \text{Con}}{\begin{array}{l}[P_1(s_1,s_2) \land s_1 \neq [] \land s_1 = h_1 :: t_1 \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\\ s_3 := T(t_1,s_2); v := h_1 :: s_3\\ [{\color{red}F} \land P_1(s_1,s_2) \land Q_1(v) \land \neg(\exists u, mem(t_1,u) \land mem(s_2,u))]\end{array}}\ \text{Seq}$$

where:

$$
\begin{aligned}
&\phi_a \equiv \\
&\quad P_2(t_1, s_2) \wedge s_1 = h_1 :: t_1 \wedge \neg mem(t_1, h_1) \wedge mem(s_2, h_1) \wedge \neg(\exists u, mem(t_1, u) \wedge mem(s_2, u)) \implies \\
&\quad P_1(s_1, s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \wedge \neg(\exists u, mem(t_1, u) \wedge mem(s_2, u)) \\
&\phi_b \equiv \\
&\quad F \wedge P_1(s_1, s_2) \wedge Q_1(v) \wedge \neg(\exists u, mem(t_1, u) \wedge mem(s_2, u)) \implies \\
&\quad P_1(s_1, s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \wedge \neg(\exists u, mem(t_1, u) \wedge mem(s_2, u)) \wedge Q_1(s_3) \wedge v = h_1 :: s_3
\end{aligned}
$$

Let's focus on ϕ_b, if stack t_1 and stack s_2 does not contain same element (as $\neg(\exists u, mem(t_1, u) \wedge mem(s_2, u))$), but s_1 and s_2 does (as P_1), it means the the shared element has to be h_1:

$$mem(s_1, h_1) \wedge mem(s_2, h_1)$$

It is easy to check that these two implication are valid.

The proof of $Spec(T, (P_2, Q_2))$ is similar, but we do not need the help of (P_1, Q_2):

$$
\dfrac{
\dfrac{
\dfrac{Spec(T,(P_1,Q_1))}{\begin{array}{l}[P_2(t_1,s_2) \wedge s_1 = h_1 :: t_1 \wedge \neg mem(t_1,h_1) \wedge \neg mem(s_2,h_1)] \\ s_3 := T(t_1,s_2) \\ [P_2(t_1,s_2) \wedge Q_2(s_3) \wedge s_1 = h_1 :: t_1 \wedge \neg mem(t_1,h_1) \wedge \neg mem(s_2,h_1)]\end{array}}\ \text{Apply-Frame} \qquad \phi_a
}{\begin{array}{l}[\ldots] \\ s_3 := T(t_1,s_2) \\ [\ldots \wedge Q_2(s_3))]\end{array}}\ \text{Cons}
\qquad
\dfrac{
\dfrac{}{\begin{array}{l}[\ldots] \\ v := h_1 :: s_3 \\ [.. \wedge v = h_1 :: s_3]\end{array}}\ \text{Assign} \qquad \phi_b
}{\begin{array}{l}[\ldots \wedge Q_2(s_3))] \\ v := h_1 :: s_3 \\ [\ldots \wedge Q_2(v)]\end{array}}\ \text{Con}
}{\begin{array}{l}[P_2(s_1,s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1] \\ s_3 := T(t_1,s_2); v := h_1 :: s_3 \\ [F \wedge P_2(s_1,s_2) \wedge Q_2(v)]\end{array}}\ \text{Seq}
$$

where

$$
\begin{aligned}
&F \equiv \\
&\quad s_1 = h_1 :: t_1 \wedge v = h_1 :: s_3 \\
&\phi_a \equiv \\
&\quad P_2(t_1, s_2) \wedge s_1 = h_1 :: t_1 \wedge \neg mem(t_1, h_1) \wedge \neg mem(s_2, h_1) \implies \\
&\quad P_2(s_1, s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \\
&\phi_b \equiv \\
&\quad F \wedge P_2(s_1, s_2) \wedge Q_2(v) \implies \\
&\quad P_2(s_1, s_2) \wedge s_1 \neq [] \wedge s_1 = h_1 :: t_1 \wedge Q_2(s_3) \wedge v = h_1 :: s_3
\end{aligned}
$$

It is easy to check that these two implication are valid.

1.6 Single inductive invariant

Can we merge $[P_1]T[Q_1]$ and $[P_2]T[Q_2]$ as a single triple? If we simply apply the disjunction rule, we have:

$$\dfrac{[P_1]C[Q_1] \qquad [P_2]C[Q_2]}{[P_1 \vee P_2]C[Q_1 \vee Q_2]}\ \text{Disjunction}$$

where $P_1 \vee P_2$ and $Q_1 \vee Q_2$ exactly equal to Σ and $\neg\Phi$ which is proved not even a underapproximation triple. The problem is that, the disjoint rule approximates too much, especially at the postcondition side (as it should merge Q_1 and Q_2, but also rules out the unreachable states).

Now we extend the postcondition Q not only describe v but also s_1 and s_2. Now we have:

$$
\begin{aligned}
&P_1 \equiv \neg dup(s_1) \wedge \neg dup(s_2) \wedge \exists u, mem(s_1, u) \wedge mem(s_2, u) \\
&Q_1 \equiv \neg dup3(v) \wedge dup(v) \\
&[P_1] v = T(s_1, s_2) [P_1 \wedge Q_1] \\
&P_2 \equiv \neg dup(s_1) \wedge \neg dup(s_2) \wedge \forall u, \neg mem(s_1, u) \vee \neg mem(s_2, u) \\
&Q_2 \equiv \neg dup3(v) \wedge \neg dup(v) \\
&[P_2] v = T(s_1, s_2) [P_2 \wedge Q_2]
\end{aligned}
$$

The a single inductive invariant can be:

$$[P_1 \vee P_2] v = T(s_1, s_2) [(P_1 \wedge Q_1) \vee (P_2 \wedge Q_2)]$$

that is:

$$
\begin{aligned}
&[\neg dup(s_1) \wedge \neg dup(s_2) \wedge \exists u, mem(s_1, u) \wedge mem(s_2, u)] \\
&v = T(s_1, s_2) \\
&[\neg dup3(v) \wedge ((\exists u, mem(s_1, u) \wedge mem(s_2, u)) \implies dup(v)) \wedge \\
&(\neg(\exists u, mem(s_1, u) \wedge mem(s_2, u)) \implies \neg dup(v))]
\end{aligned}
$$

ACKNOWLEDGEMENTS

REFERENCES