Addendum to: The Bane of Generate-and-Validate Program Repair: Too Much Generation, Too Little Validation

Anonymous, Pending Review
[REG] Regular Research Paper

***Abstract*—Sample Hoare-style proof of program repair alagorithm.**

***Index Terms*—program repair, absolute correctness, relative correctness, patch generation, patch validation, Cardumen**

We propose to prove the partial correctness of `UnitIncCor()` with respect to the second specification. This amounts to proving that the following Hoare formula is a theorem in Hoare's deductive logic:

v: $\{(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P)\}$

```
m=1; inc=false; Pp=P;
while (! inc && m<=M)
  {while (! smc(Pp,P) && MorePatches(P,m))
    {Pp = NextPatch(P,m);}
   if smc(Pp,P) {inc=true;}
   else {m=m+1;}}//try higher multiplicity
```

$\{Pp \sqsupset_{R'} P\}$.

Applying the sequence rule to v, with the following intermediate predicate int:

$$(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P)$$

$$\wedge m = 1 \wedge inc \wedge Pp = P$$

yields the following lemmas:

v_0: $\{(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P)\}$

```
m=1; inc=false; Pp=P;
```

$\{(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P) \wedge m = 1 \wedge \neg inc \wedge Pp = P\}$.

v_1: $\{(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P) \wedge m = 1 \wedge \neg inc \wedge Pp = P\}$

```
while (! inc && m<=M)
  {while (! smc(Pp,P) && MorePatches(P,m))
    {Pp = NextPatch(P,m);}
   if smc(Pp,P) {inc=true;}
   else {m=m+1;}}//try higher multiplicity
```

$\{Pp \sqsupset_{R'} P\}$.

If we apply the (concurrent) assignment rule to v_0, we get:

v_{00}: $(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P)$
$\Rightarrow$
$(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P) \wedge 1 = 1 \wedge \textbf{true} \ \wedge P = P\}$.

This formula is clearly a tautology, hence we turn our attention to v_1, to which we apply the while rule, with the following loop invariant inv:

$$inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P)$$

$$\vee(\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))),$$

where $inb(m)$ (stands for: *in bounds*) is shorthand for: $1 \leq m \leq M$. Application of the while rule to v_1 with the selected loop invariant yields three lemmas:

v_{10}: $(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P) \wedge m = 1 \wedge \neg inc \wedge Pp = P$
$\Rightarrow$
$inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))$.

v_{11}: $\{(\neg inc \wedge m \leq M) \wedge inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))\}$

```
{while (! smc(Pp,P) && MorePatches(P,m))
  {Pp = NextPatch(P,m);}
 if smc(Pp,P) {inc=true;}
 else {m=m+1;}}//try higher multiplicity
```

$\{inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))\}$.

v_{12}: $\neg(\neg inc \wedge m \leq M) \wedge inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))$
$\Rightarrow$
$Pp \sqsupset_{R'} P$.

To check the validity of v_{10}, we rewrite it by distributing $inb(m)$ over the disjunction and replacing m by 1 on the right hand side:

v_{10}: $(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P) \wedge m = 1 \wedge \neg inc \wedge Pp = P$
$\Rightarrow$
$(inb(m) \wedge inc \wedge Pp \sqsupset_{R'} P) \vee (inb(m) \wedge \neg inc \wedge (\exists h : 1 \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$.

Now it is clear that v_{10} is a tautology, since the left hand side logically implies the second disjunct of the right hand side, assuming, as we do, that $M \geq 1$. As for v_{12}, its left hand side can be simplified into $(inc \wedge Pp \sqsupset_{R'} P)$, due to the contradiction between $m > M$ and $inb(m)$, and the contradiction between inc and $\neg inc$. Hence v_{12} is also a tautology. We turn our attention to v_{11}, which we first simplify as follows:

v_{11}: $\{\neg inc \wedge inb(m) \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)\}$

```
{while (! smc(Pp,P) && MorePatches(P,m))
  {Pp = NextPatch(P,m);}
 if smc(Pp,P) {inc=true;}
 else {m=m+1;}}//try higher multiplicity
```

$\{inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))\}$.

We apply the sequence rule to v_{11}, with the following intermediate predicate int':

$$(Pp \sqsupset_{R'} P \vee PS(m) = \epsilon) \wedge$$

$$\neg inc \wedge inb(m) \wedge$$

$$(Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)).$$

This yields the following two lemmas:

v_{110}: $\{\neg inc \wedge inb(m) \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))\}$

```
{while (! smc(Pp,P) && MorePatches(P,m))
  {Pp = NextPatch(P,m);}
```

$\{(Pp \sqsupset_{R'} P \vee PS(m) = \epsilon) \wedge \neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)).\}$.

v_{111}: $\{(Pp \sqsupset_{R'} P \vee PS(m) = \epsilon) \wedge \neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)).\}$

```
if smc(Pp,P) {inc=true;}
else {m=m+1;}}//try higher multiplicity
```

$\{inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))\}$.

We apply the while rule to v_{110}, with the following loop invariant, inv':

$$\neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$$

This yields the following three lemmas:

v_{1100}: $\neg inc \wedge inb(m) \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$
$\Rightarrow$
$\neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$.

v_{1101}: $\{\neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$
$\wedge \neg(Pp \sqsupset_{R'} P \wedge PS(m) \neq \epsilon)\}$

```
{Pp = NextPatch(P,m);}
```

$\{\neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))\}$

v_{1102}: $\neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)) \wedge (Pp \sqsupset_{R'} P \vee PS(m) = \epsilon)$
$\Rightarrow$
$(Pp \sqsupset_{R'} P \vee PS(m) = \epsilon) \wedge \neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$.

To see that v_{1100} is a tautology, it suffices to distribute the $\wedge$ over the $\vee$ on the right hand side of the implication, and to notice that the second disjunct on the right hand side is a copy of the left hand side of the implication. As for v_{1102}, it is clearly a tautology, since the right hand side of $\Rightarrow$ is merely a copy of the left hand side. We turn our attention to v_{1101} now, and we begin by simplifying its precondition by virtue of Boolean identities:

v_{1101}: $\{\neg inc \wedge inb(m) \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P) \wedge \neg(Pp \sqsupset_{R'} P) \wedge PS(m) \neq \epsilon)\}$

```
{Pp = NextPatch(P,m);}
```

$\{\neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))\}$

We consider v_{1101}, to which we must apply the assignment statement rule; to this effect, we must analyze the semantics of function `NextPatch(P,m)`. We assume that this function performs the following operations:

```
Pp=head(PS(m)); PS(m)=tail(PS(m));
```

Hence application of the assignment rule yields the following formula:

v_{11010}: $\neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$
$\wedge (\neg(Pp \sqsupset_{R'} P \wedge PS(m) \neq \epsilon)$
$\Rightarrow$
$\neg inc \wedge inb(m) \wedge (head(PS(m)) \sqsupset_{R'} P \vee (\exists Q \in tail(PS(m)) : Q \sqsupset_{R'} P) \vee (\exists h : m+1 \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P))$.

We consider the first two disjuncts in the parenthesized expression:
$(head(PS(m)) \sqsupset_{R'} P) \vee (\exists Q \in tail(PS(m)) : Q \sqsupset_{R'} P)$
and we merge them into a single expression:
$(\exists Q \in PS(m) : Q \sqsupset_{R'} P)$.
Now we merge this expression with the third disjunct above:
$(\exists Q \in PS(m) : Q \sqsupset_{R'} P) \vee (\exists h : m+1 \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)$,
to obtain:
$(\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)$.
Replacing these in v_{11010}, we find that the right hand side is a logical conclusion of the left hand side, hence v_{11010} is a tautology. We now consider v_{111}, to which we apply the if-then-else rule, which yields two lemmas:

v_{1110}: $\{(Pp \sqsupset_{R'} P) \wedge (Pp \sqsupset_{R'} P \vee PS(m) = \epsilon) \wedge \neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)).\}$

```
inc=true;
```

$\{inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))\}$.

v_{1111}: $\{\neg(Pp \sqsupset_{R'} P) \wedge (Pp \sqsupset_{R'} P \vee PS(m) = \epsilon) \wedge \neg inc \wedge inb(m) \wedge (Pp \sqsupset_{R'} P \vee (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)).\}$

```
m=m+1;
```

$\{inb(m) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))\}$.

We simplify v_{1110} and apply the assignment rule to it, yielding:

v_{11100}: $(Pp \sqsupset_{R'} P) \wedge \neg inc \wedge inb(m) \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)$
$\Rightarrow$
$inb(m) \wedge (Pp \sqsupset_{R'} P)$,

This is clearly a tautology. We simplify v_{1111} and apply the assignment rule to it, yielding:

v_{11110}: $\neg(Pp \sqsupset_{R'} P) \wedge PS(m) = \epsilon \wedge \neg inc \wedge inb(m) \wedge (\exists h : m \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)$
$\Rightarrow$
$inb(m+1) \wedge ((inc \wedge Pp \sqsupset_{R'} P) \vee (\neg inc \wedge (\exists h : m+1 \leq h \leq M : \exists Q \in PS(h) : Q \sqsupset_{R'} P)))\}$.

If we know that there exists Q strictly more-correct than P in one of the patch sequences $PS(m), PS(m+1), ...PS(M)$ but $PS(m)$ is empty, then it must be in one of the sequence $PS(m+1), PS(m+2), ...PS(M)$. For the same reason, m is necessarily strictly less than M, since Q is somewhere in $PS(m+1), PS(m+2), ...PS(M)$. Hence $inb(m+1)$ holds. We conclude that v_{11110} is a tautology.

Since all the lemmas generated form v are valid, so is v. Hence `UnitIncCor()` is partially correct with respect to the specification:

- Precondition: $(\exists m : 1 \leq m \leq M : \exists Q \in PS(m) : Q \sqsupset_{R'} P)$.
- Postcocndition: $Pp \sqsupset_{R'} P$.