\[ [\text{skip}]_\text{ok} = \{(s, s) \mid s \in \text{World}\} \quad [\text{skip}]_\text{er} = \emptyset \]

\[ [x := e]_\text{ok} = \{(s, (\eta, (\eta[x \rightarrow v], \sigma))) \mid [e]_\eta = v\} \quad [x := e]_\text{er} = \emptyset \]

\[ [\text{assume}(B)]_\text{ok} = \{(s, s) \mid s = (\eta, \sigma) \land [B]_\eta \neq 0\} \quad [\text{assume}(B)]_\text{er} = \emptyset \]

\[ [\text{error()}]_\text{ok} = \emptyset \quad [\text{error()}]_\text{er} = \{(s, s) \mid s \in \text{World}\} \]

\[ [C_1; C_2]_\epsilon = \left\{(s, s') \mid \epsilon = \text{error} \land (s, s') \in [C_1]_\epsilon \lor \exists s'' \cdot (s, s'') \in [C_1]_\text{ok} \land (s'', s') \in [C_2]_\epsilon \right\} \]

\[ [C_1 + C_2]_\epsilon = [C_1]_\epsilon \cup [C_2]_\epsilon \]

\[ [C^*]_\epsilon = \bigcup_{i \in \mathbb{N}} [C]^i_\epsilon \quad \text{with} \quad C^0_\epsilon = \text{skip} \quad \text{and} \quad C^{i+1}_\epsilon = C; C^i \]

\[ [x := \text{malloc}]_\text{ok} = \left\{(s, (\eta[x \rightarrow l], \sigma[l \rightarrow v])) \mid s = (\eta, \sigma) \land v \in \text{Val} \land (l \notin \text{dom } (\sigma) \lor \sigma(l) = \perp) \right\} \]

\[ \quad \cup \{(s, (\eta[x \rightarrow \text{nil}], \sigma)) \mid s = (\eta, \sigma)\} \]

\[ [x := \text{malloc}]_\text{er} = \emptyset \]

\[ [\text{free}(x)]_\text{ok} = \left\{(s, (\eta, \sigma[\eta(x) \rightarrow \perp])) \mid s = (\eta, \sigma) \land \sigma(\eta(x)) \in \text{Val}\right\} \]

\[ [\text{free}(x)]_\text{er} = \left\{(s, s) \mid s = (\eta, \sigma) \land (\eta(x) = \text{nil} \lor \sigma(\eta(x)) = \perp)\right\} \]

\[ [x := y]_\text{ok} = \left\{(s, (\eta[x \rightarrow v], \sigma)) \mid s = (\eta, \sigma) \land \sigma(\eta(y)) = v \in \text{Val}\right\} \]

\[ [x := y]_\text{er} = \left\{(s, s) \mid \sigma = (s, h(\otimes)) \land (\eta(y) = \text{nil} \lor \sigma(\eta(y)) = \perp)\right\} \]

\[ [[x] := y]_\text{ok} = \left\{(s, (\eta, \sigma[\eta(x) \rightarrow \eta(y)])) \mid s = (\eta, \sigma) \land \sigma(\eta(x)) \in \text{Val}\right\} \]

\[ [[x] := y]_\text{er} = \left\{(s, s) \mid \sigma = (\eta, \sigma) \land (\eta(x) = \text{nil} \lor \sigma(\eta(x)) = \perp)\right\} \]

\[ [\text{local } x. C]_\epsilon = \left\{((\eta, \sigma), (\eta'[x \rightarrow \eta(x)], \sigma')) \mid ((\eta[x \rightarrow \text{nil}], \sigma), (\eta', \sigma')) \in [C]_\epsilon\right\} \]

Fig. 8. The Pulse-X denotational semantics

A SEMANTICS

The Pulse-X denotational semantic is given in Fig. 8, and is analogous to that of ISL in [Raad et al. 2020].
Finding Real Bugs in Big Programs with Incorrectness Logic

B  ERROR TRACE OF NPE IN LISTING 1

1569  apps/lib/s_cb.c:959: error: Nullptr Dereference
1570  PSL found a potential null pointer dereference on line 959.
1571
1572  apps/lib/s_cb.c:957: in call to `app_malloc`
1573  955. static int ssl_excert_prepend(SSL_EXCERT **pexc)
1574  956. {
1575  957.   SSL_EXCERT *exc = app_malloc(sizeof(*exc), "prepend cert");
1576  958.   memset(exc, 0, sizeof(*exc));
1577  959.     return from call to `app_malloc`
1580
1581  test/testutil/apps_mem.c:16:16: in call to `CRYPTO_malloc` (modelled)
1582  14. void *app_malloc(size_t sz, const char *what)
1583  15. {
1584  16.   void *vp = OPENSSL_malloc(sz);
1585  17.     return vp;
1586
1587  test/testutil/apps_mem.c:16:16: is the null pointer
1589  14. void *app_malloc(size_t sz, const char *what)
1590  15. {
1591  16.   void *vp = OPENSSL_malloc(sz);
1592  17.     return vp;
1593
1595  test/testutil/apps_mem.c:16:5: assigned
1597  14. void *app_malloc(size_t sz, const char *what)
1598  15. {
1599  16.   void *vp = OPENSSL_malloc(sz);
1600  17.     return vp;
1601
1603  test/testutil/apps_mem.c:18:5: returned
1604  16.   void *vp = OPENSSL_malloc(sz);
1605  17.     return vp;
1606  18.     return vp;
1607  19. }
1608
1609  apps/lib/s_cb.c:957:23: return from call to `app_malloc`
1610  955. static int ssl_excert_prepend(SSL_EXCERT **pexc)
1611  956. {
1612  957.   SSL_EXCERT *exc = app_malloc(sizeof(*exc), "prepend cert");
1613  958.   memset(exc, 0, sizeof(*exc));
1615  959.     return from call to `app_malloc`
apps/lib/s_cb.c:957:5: assigned
static int ssl_excert_prepend(SSL_EXCERT **pexc)
{
SSL_EXCERT *exc = app_malloc(sizeof(*exc), "prepend cert");
memset(exc, 0, sizeof(*exc));
exc->next = *pexc;
}

apps/lib/s_cb.c:959:5: invalid access occurs here
SSL_EXCERT *exc = app_malloc(sizeof(*exc), "prepend cert");
memset(exc, 0, sizeof(*exc));
exc->next = *pexc;

Listing 10. Error trace of the bug in Listing 1.
C MANIFEST ERRORS

THEOREM C.1. For all manifest errors $\models [p] C [er: q]$:

$$\forall s, \exists s'. (s, s') \in [C]_{er} \land s' \in (q \ast true)$$

PROOF. Pick a valid ISL triple $\models [p] C [er: q]$ denoting a manifest error. Pick arbitrary $s$ and let $r \triangleq \{s\}$. From Def. 3.2 we then know there exists $f$ such that $p \ast f \vdash r$ and $\text{sat}(q \ast f)$ holds. As $[p] C [er: q]$ is a valid triple, using the Frame rule of ISL we can derive $\models [p \ast f] C [er: q \ast f]$. Subsequently, since $p \ast f \vdash r$, from Def. 3.1 we also have $\models [r] C [er: q \ast f]$. On the other hand, as $\text{sat}(q \ast f)$ holds, we know there exists $s'$ such that $s' \in (q \ast f)$ and thus $s' \in (q \ast true)$. Moreover, from $\models [r] C [er: q \ast f]$, Def. 3.1 and the definitions of $r$ we know $(q \ast f) \subseteq [C]_{er}(\{s\})$; i.e., $\forall s_q \in (q \ast f)$. $(s, s_q) \in [C]_{er}$, and thus $(s, s') \in [C]_{er}$, as required.

\[\square\]

Manifest Errors and Reverse Under-Approximate Triples. We next demonstrate that under certain conditions manifest errors coincide with reverse under-approximate triples. We write $\models \{p\} C \{e : q\}$ to denote a reverse (under-approximate) triple that is valid. Intuitively, a valid reverse triple is the dual of a valid ISL triple (Def. 3.1): $\models \{p\} C \{e : q\}$ denotes that executing $C$ on each state in $p$ reaches some state in $q$ under $e$, while preserving the frame property. Put formally:

$$\models \{p\} C \{e : q\} \overset{\text{def}}{\iff} \forall r. (p \ast r) \subseteq [C]_{e}^{-1}((q \ast r))$$

where $[C]_{e}^{-1}$ denotes the inverse of the $[C]_{e}$ relation in Def. 3.1. The notion of reverse under-approximate triples corresponds to that of total Hoare triples in [de Vries and Koutavas 2011].

Interestingly, as we show in Theorem C.2 below, given a manifest error $T \triangleq \models [p] C [er: q]$, if $p \equiv \text{emp} \land \text{true}$ (i.e., $p$ imposes no spatial or pure constraints on the context), then $T$ is also a valid reverse triple, i.e., $\models \{p\} C \{er : q\}$ also holds.

THEOREM C.2. Given a manifest error $\models [p] C [er: q]$, if $p \equiv \text{emp} \land \text{true}$, then $\models \{p\} C \{er : q\}$.

PROOF. Pick a manifest error $T \triangleq \models [p] C [er: q]$ such that $p \equiv \text{emp} \land \text{true}$. Pick arbitrary $r$ and $s \in (p \ast r)$; it then suffices to show there exists $s' \in (q \ast r)$ such that $(s, s') \in [C]_{e}$.

Let $r' \triangleq \{s\};$ as $\text{sat}(r')$ holds and $T$ denotes a manifest error, from Def. 3.2 we know $\text{sat}(q \ast r')$ holds. Moreover as $s \in (p \ast r)$ and $p \equiv \text{emp} \land \text{true}$ (and thus from the semantics of assertions $p \ast r \equiv r$), we know $s \in (r')$ and thus $r' \vdash r$. Moreover, as $\text{sat}(q \ast r')$ holds, we know there exists $s'$ such that $s' \in (q \ast r')$ and thus $s' \in (q \ast r)$ since $r' \vdash r$. On the other hand, as $T$ is a valid triple, from Def. 3.1 we know $(q \ast r') \subseteq [C]_{e}((q \ast r'))$ and thus $(q \ast r') \subseteq [C]_{e}((r'))$ since $p \equiv \text{emp} \land \text{true}$. Consequently, as $s' \in (q \ast r')$, we know $s' \in [C]_{e}(r')$, i.e., $(s, s') \in [C]_{e}$ since $r' \triangleq \{s\}$. That is, there exists $s' \in (q \ast r)$ such that $(s, s') \in [C]_{e}$, as required.

\[\square\]

Definition C.3 (Path-manifest error). An error triple $\models [p] C [er: q]$ denotes a path-manifest error iff for all $r$, if $\text{sat}(p \ast r)$ holds then $\text{sat}(q \ast r)$ also holds.

THEOREM C.4 (Path-manifest errors). An error triple $\models [p] C [er: q]$ with $p \equiv \exists X_p. \kappa_p \land \pi_p$ and $q \equiv \exists X_{q}. \kappa_q \land \pi_q$ denotes a path-manifest error if:

1. $\text{sat}(q)$ holds;
2. $\text{locs}(\kappa_q) \setminus X_q \subseteq \text{locs}(\kappa_p) \setminus X_p$;
3. for all $\overrightarrow{v}$, $\text{sat}(\pi_q[\overrightarrow{v} / \overrightarrow{Y} \cup \text{locs}(\kappa_q)])$ holds, where $\overrightarrow{Y} = \text{flv}(q)$ and:
4. $\text{locs}(\text{emp}) \triangleq \emptyset \quad \text{locs}(X \rightarrow X) \triangleq \{X\} \quad \text{locs}(X \rightarrow V) = \text{locs}(X \rightarrow Y) \triangleq \{X\} \quad \text{locs}(\kappa_1 \ast \kappa_2) \triangleq \text{locs}(\kappa_1) \cup \text{locs}(\kappa_2)$
Proof. Pick arbitrary $p$, $q$, $r$, $C$, $\overrightarrow{X_p}$, $\kappa_p$, $\pi_p$, $\overrightarrow{X_q}$, $\kappa_q$, $\pi_q$ and $\overrightarrow{Y}$ such that $p \models \exists X_p$. $\kappa_p \land \pi_p$, $q \models \exists X_q$. $\kappa_q \land \pi_q$, $\overrightarrow{Y} = \text{flv}(q)$, sat($q$) holds, $\text{locs}(\kappa_q) \setminus \overrightarrow{X_q} \subseteq \text{locs}(\kappa_p) \setminus \overrightarrow{X_p}$, for all $\overrightarrow{\sigma}$, sat($\pi_q[\overrightarrow{\sigma}/\overrightarrow{Y} \cup \text{locs}(\kappa_q)]$) holds and sat($p \land r$) holds.

As sat($p \land r$) holds, we know there exist $\eta$, $\eta_p$, $\sigma_p$, $\sigma_r$, $\overrightarrow{\nu}$ such that $\eta_p = \eta[\overrightarrow{X_p} \mapsto \overrightarrow{\nu}]$, $\eta_p, \sigma_p \models \kappa_p$, $\eta_p \models \pi_p$, $\eta_r, \sigma_r \models r$ and $\sigma_p \neq \sigma_r$, i.e., $(\eta(\text{locs}(\kappa_r)) \setminus \overrightarrow{X_p}) \cap \text{dom}(\sigma_r) = \emptyset$.

As $\text{locs}(\kappa_q) \setminus \overrightarrow{X_q} \subseteq \text{locs}(\kappa_p) \setminus \overrightarrow{X_p}$, we know there exist $\overrightarrow{Z_1}, \overrightarrow{Z_2}$ such that $\text{locs}(\kappa_q) = \overrightarrow{Z_1} \cup \overrightarrow{Z_2}$, $\overrightarrow{Z_1} \cap \overrightarrow{X_q} = \emptyset$ (i.e., $\overrightarrow{Z_1} \subseteq \overrightarrow{Y}$), $\overrightarrow{Z_1} \subseteq \text{locs}(\kappa_p) \setminus \overrightarrow{X_p}$ and $\overrightarrow{Z_2} \subseteq \overrightarrow{X_q}$. Note that as $\overrightarrow{Z_1} \subseteq \text{locs}(\kappa_p) \setminus \overrightarrow{X_p}$ and $\eta_p, \sigma_p \models \kappa_p$, from the definition of $\eta$ we know that $\eta \models \text{disjoint}(\overrightarrow{Z_1})$ holds.

Pick $\overrightarrow{\nu_2}$ such that $\overrightarrow{\nu_2} \cap \text{dom}(\sigma_r) = \emptyset$ and $\overrightarrow{\nu_2} \cap \eta(\overrightarrow{Z_1}) = \emptyset$. Let $\pi_1 = \pi[\overrightarrow{Y} \mapsto \overrightarrow{\nu}]$ and $\pi_2 = \pi_1[\overrightarrow{Z_2} \mapsto \overrightarrow{\nu_2}]$. As for all $\overrightarrow{\sigma}$, sat($\pi_q[\overrightarrow{\sigma}/\overrightarrow{Y} \cup \text{locs}(\kappa_q)]$) holds, we know that sat($\pi_2$) holds and thus there exists $\eta_2$, $\overrightarrow{\nu_3}$ such that $\eta_2 = \eta[\overrightarrow{Z_2} \mapsto \overrightarrow{\nu_3}][\overrightarrow{X_q} \setminus \overrightarrow{Z_2}]$ and $\eta_2 \models \pi_2$. That is, there exist $\overrightarrow{\nu_2}$ such that $\eta_2 = \eta[\overrightarrow{X_q} \mapsto \overrightarrow{\nu_2}]$, $\eta_2(\overrightarrow{Y}) = \eta(\overrightarrow{Y})$ and $\eta(\overrightarrow{Z_2}) = \overrightarrow{\nu_2}$. Moreover, since $\eta \models \text{disjoint}(\overrightarrow{Z_1})$, $\overrightarrow{Z_1} \subseteq \overrightarrow{Y}$, $\eta(\overrightarrow{Y}) = \eta(\overrightarrow{Y})$, $\overrightarrow{Z_1} \cap \eta(\overrightarrow{Z_1}) = \emptyset$ and $\eta(\overrightarrow{Z_2}) = \overrightarrow{\nu_2}$, we also have $\eta \models \text{disjoint}(\overrightarrow{Z_1} \cup \overrightarrow{Z_2})$. As such, since $\text{locs}(\kappa_q) = \overrightarrow{Z_1} \cup \overrightarrow{Z_2}$, from Proposition C.9 we have $\eta_q, \sigma_q \models \kappa_q$, where $\sigma_q = [\kappa_q]_{\eta_q}$. Consequently, as $\eta = \eta[\overrightarrow{X_q} \mapsto \overrightarrow{\nu_2}]$, $\eta_q, \sigma_q \models \kappa_q$ and $\eta_q \models \pi_q$ we have $\eta, \sigma_q \models \exists X_q$. $\kappa_q \land \pi_q$ and thus $\eta, \sigma_q \models q$.

Lastly, since $\text{dom}(\sigma_q) = \eta_q(\overrightarrow{Z_1}) \cup \eta_q(\overrightarrow{Z_2}) = \eta(\overrightarrow{Z_1}) \cup \overrightarrow{\nu_2} \cap \text{dom}(\sigma_r) = \emptyset$, $(\eta(\text{locs}(\kappa_r)) \setminus \overrightarrow{X_p}) \cap \text{dom}(\sigma_r) = \emptyset$ and thus $\eta(\overrightarrow{Z_1}) \cap \text{dom}(\sigma_r) = \emptyset$ (since $\overrightarrow{Z_1} \subseteq \text{locs}(\kappa_p) \setminus \overrightarrow{X_p}$), we also know that $\text{dom}(\sigma_r) = \emptyset$ and $\sigma_r \neq \sigma_r$. Consequently, as $\eta, \sigma_q \models q$, $\eta, \sigma_r \models r$ and $\sigma_q \neq \sigma_r$, we know $\eta, \sigma_q \cup \sigma_r \models q \land r$ and thus sat($q \land r$), as required. 

Definition C.5 (Resource-manifest errors). An error triple $\models [p] C [er: q]$ denotes a resource-manifest error iff:

- $\models [p] C [er: q]$ denotes a path-manifest error; and
- $\text{pheap}(p)$ holds, where

\[
\text{pheap}(\exists X. \kappa \land \pi) \overset{\text{def}}{\iff} \text{nllocs}(\kappa) = \emptyset \land \pi = \text{true}
\]

\[
\text{nllocs}(\text{emp}) \overset{\text{def}}{=} \emptyset \quad \text{nllocs}(X \to V) \overset{\text{def}}{=} \emptyset \quad \text{nllocs}(X \to Y) \overset{\text{def}}{=} \emptyset
\]

\[
\text{nllocs}(X \to V) \overset{\text{def}}{=} \emptyset \quad \text{nllocs}(X \to Y) \overset{\text{def}}{=} \emptyset
\]

\[
\text{nllocs}(\kappa_1 \ast \kappa_2) = \text{nllocs}(\kappa_1) \cup \text{nllocs}(\kappa_2)
\]

Theorem C.6 (Resource-manifest errors). An error triple $\models [p] C [er: q]$ with $p \models \exists X_q. \kappa_q \land \pi_q$ and $q \models \exists X_q. \kappa_q \land \pi_q$ denotes a resource-manifest error if:

(1) sat($q$) holds;

(2) $\text{locs}(\kappa_q) \setminus \overrightarrow{X_q} \subseteq \text{locs}(\kappa_p) \setminus \overrightarrow{X_p}$;

(3) for all $\overrightarrow{\sigma}$, sat($\pi_q[\overrightarrow{\sigma}/\overrightarrow{Y} \cup \text{locs}(\kappa_q)]$) holds, where $\overrightarrow{Y} = \text{flv}(q)$;

(4) $\text{pheap}(p)$ holds.

Proof. Follows from Theorem C.4 and the definitions of $\text{pheap}(.)$ and resource-manifest errors.

Definition C.7 (Manifest errors). An error triple $\models [p] C [er: q]$ denotes a manifest error iff for all $r$, if sat($r$) holds, then there exists $f$ such that $p \land f \land r$ and sat($q \land f$) also holds.

Theorem C.8 (Manifest errors). An error triple $\models [p] C [er: q]$ with $p \models \exists X_q. \kappa_q \land \pi_q$ denotes a manifest error if:
Finding Real Bugs in Big Programs with Incorrectness Logic

(1) $p \equiv \text{emp} \land \text{true};$

(2) $\text{sat}(q)$ holds;

(3) $\text{locs}(\kappa_q) \setminus X_q \subseteq \text{locs}(\kappa_p) \setminus X_p;$

(4) for all $\mathcal{O}$, $\text{sat}(\pi_q[\mathcal{O}/Y \cup \text{locs}(\kappa_q)])$ holds, where $\overrightarrow{Y} = \text{flv}(q)$.

**Proof.** Pick an arbitrary error triple $\models [p] \subseteq [\text{er}: q]$ such that conditions (1)-(4) above hold. Pick an arbitrary $r$ such that $\text{sat}(r)$ holds. As $p \equiv \text{emp} \land \text{true}$, we then have $p * r \equiv r$, and thus $p * r \vdash r$.

Moreover, from conditions (2)-(4) and Theorem C.4 we know $\models [p] \subseteq [\text{er}: q]$ is a path-manifest error. On the other hand, as $\text{sat}(r)$ holds and $p * r \equiv r$, we know $\text{sat}(p * r)$ and thus from the definition of path-manifest errors we have $\text{sat}(q * r)$, as required.

**Proposition C.9.** For all $\kappa$, $\pi$, $\eta$, $\sigma$, $\sigma_1$, $\sigma_2$, $\kappa_1$, $\kappa_2$:

(1) if $\eta, \sigma_1 \models \kappa$ and $\eta, \sigma_2 \models \kappa$, then $\sigma_1 = \sigma_2$;

(2) $\kappa \equiv \kappa \land \text{disjoint}(\text{locs}(\kappa))$, where:

\[
\text{disjoint}(\emptyset) = \text{disjoint}(\{X\}) \triangleq \text{true} \quad \text{disjoint}(\{X\} \cup S) \triangleq \bigwedge_{Y \in S} X \neq Y \land \text{disjoint}(S)
\]

(3) if $\eta, \sigma \models \kappa$, then $\eta \models \text{disjoint}(\text{locs}(\kappa))$ (follows from the previous part);

(4) if $\eta \models \text{disjoint}(\text{locs}(\kappa))$, then $\eta, [\kappa]_\eta \models \kappa$, where:

\[
[\text{emp}]_\eta = \emptyset \quad [X \rightarrow X]_\eta = [\eta(x) \mapsto \eta(V)] \quad [X \rightarrow Y]_\eta = [\eta(X) \mapsto \eta(Y)]
\]

\[
[X \not\rightarrow]_\eta = [\eta(X) \mapsto \bot] \quad [\kappa_1 \star \kappa_2]_\eta = [\kappa_1]_\eta \lor [\kappa_2]_\eta
\]

(5) if $\eta, \sigma \models \kappa$, then $\sigma = [\kappa]_\eta$ (follows from parts 1, 3 and 4).