ESVerify Theory: Definitions, Axioms and Theorems

1 Logic

$$
\begin{array}{rll}
P \in \text{Propositions} & ::= & t \mid \neg P \mid P \wedge P \mid P \vee P \mid pre_1(\otimes, t) \mid pre_2(\oplus, t, t) \mid pre(t, t) \mid \\
 & & post(t, t) \mid call(t) \mid \forall x.\{call(x)\} \Rightarrow P \mid \exists x.P \\
t \in \text{Terms} & ::= & v \mid x \mid \otimes t \mid t \oplus t \mid t(t) \\
\otimes \in \text{UnaryOperators} & ::= & \neg \mid isInt \mid isBool \mid isFunc \\
\oplus \in \text{BinaryOperators} & ::= & + \mid - \mid \times \mid / \mid \wedge \mid \vee \mid = \mid < \\
v \in \text{Values} & ::= & \text{true} \mid \text{false} \mid n \mid \langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle \\
\sigma \in \text{Environments} & ::= & \emptyset \mid \sigma[x \mapsto v] \\
f, x, y, z \in \text{Variables} & & \\
n \in \mathbb{N} & &
\end{array}
$$

The meaning of unary and binary operators is specified by the partial function δ, e.g. $\delta(+, 2, 3) = 5$. Validity of logical propositions is axiomatized as follows.

Notation 1 (Implication). $P \Rightarrow Q \quad \equiv \quad \neg P \vee Q$.

Axiom 1 (L-True). $\vdash$ true.

$\boxed{\vdash P}$

Axiom 2 (L-And). Iff $(\vdash P)$ and $(\vdash Q)$ then $(\vdash P \wedge Q)$.

Axiom 3 (L-Or-Left). If $(\vdash P)$ then $\vdash P \vee Q$.

Axiom 4 (L-Or-Right). If $(\vdash Q)$ then $\vdash P \vee Q$.

Axiom 5 (L-Or-Elim). If $(\vdash P \vee Q)$ then $\vdash P$ or $\vdash Q$.

Axiom 6 (L-No-Contradictions). $(\vdash P \wedge \neg P)$ is not true.

Axiom 7 (L-Excluded-Middle). $(\vdash P \vee \neg P)$.

Axiom 8 (L-Terms). Iff $(\vdash t)$ then $(\vdash t = true)$.

Axiom 9 (L-Forall). If $(\vdash P[x \mapsto v])$ for all values v, then $\vdash \forall x.P$.

Axiom 10 (L-Implicit-Universally-Quantified). If x is free in P and $(\vdash P)$ then $\vdash \forall x.P$.

Axiom 11 (L-Forall-Elim). If $\vdash \forall x.P$ then $\vdash P[x \mapsto t]$ for all terms t.

Axiom 12 (L-Unary-Op). Iff $\delta(\otimes, v_x) = v$ then $\vdash v = \otimes v_x$.

Axiom 13 (L-Binary-Op). Iff $\delta(\oplus, v_x, v_y) = v$ then $\vdash v = v_x \oplus v_y$.

Axiom 14 (L-Unary-Op-Pre). If $\vdash pre_1(\otimes, v_x)$ then $(\otimes, v_x) \in dom(\delta)$.

Axiom 15 (L-Binary-Op-Pre). If $\vdash pre_2(\oplus, v_x, v_y)$ then $(\oplus, v_x, v_y) \in dom(\delta)$.

Definition 1 (Lookup). $\sigma(x)$ looks up x is the environment σ.

Definition 2 (Substitution). $\sigma(t)$ and $\sigma(P)$ substitute free variables in t and P with values from σ.

Notation 2 (Models). $\sigma \models P \quad \equiv \quad \vdash \sigma(P)$.

$\boxed{\sigma \vdash P}$

2 Quantifier Instantiation Algorithm

The following quantifier instantiation algorithm manipulates propositions. The resulting propositions are quantifier-free and checked with an SMT solver.

$$\boxed{\langle P \rangle}$$

$$\begin{array}{ll} P^+[\circ] ::= & \circ \mid \neg P^-[\circ] \mid P^+[\circ] \wedge P \mid P \wedge P^+[\circ] \mid P^+[\circ] \vee P \mid P \vee P^+[\circ] \\ P^-[\circ] ::= & \neg P^+[\circ] \mid P^-[\circ] \wedge P \mid P \wedge P^-[\circ] \mid P^-[\circ] \vee P \mid P \vee P^-[\circ] \end{array}$$

$$\begin{array}{lll} calls^+\ (P) & := & \{\ call(t) \mid P = P^+[call(t)]\ \} \\ calls^-\ (P) & := & \{\ call(t) \mid P = P^-[call(t)]\ \} \end{array}$$

$$\begin{array}{llll} lift^+\ (P) & := & \text{match } P \text{ with} & \\ & & P^-[\exists x. P'] \quad \rightarrow lift^+\ (P^-[P'[x \mapsto y]]) & (y \text{ fresh}) \\ & & P^+[\forall x. \{call(x)\} \Rightarrow P'] \rightarrow lift^+\ (P^+[call(y) \Rightarrow P'[x \mapsto y]]) & (y \text{ fresh}) \\ & & \text{otherwise} \quad \rightarrow P & \end{array}$$

$$erase^-\ (P) \ := P\left[P^-\left[(\forall x. \{call(x)\} \Rightarrow P')\right] \mapsto P^-\left[\text{true}\right], P^-\left[call(t)\right] \mapsto P^-\left[\text{true}\right]\right]$$

$$instOnce^-\ (P) \ :=$$

$$P\left[P^-\left[(\forall x. \{call(x)\} \Rightarrow P')\right] \mapsto P^-\left[(\forall x. \{call(x)\} \Rightarrow P') \wedge \bigwedge_{call(t) \in calls^-(P)} P'[x \mapsto t]\right]\right]$$

$$\begin{array}{lll} inst^-\ (P, 0) & := inst^-\ \text{is}\ erase^-(lift^+(P)) & \\ inst^-\ (P, n) & := inst^-(instOnce^-\ (lift^+(P)), n-1) & (n > 0) \end{array}$$

$$\begin{array}{ll} \langle P \rangle \ := & \text{let } n = \text{maximum level of quantifier nesting of } lift^+(P) \\ & \text{in } \neg Sat\ (\neg inst^-\ (P, n)) \end{array}$$

Definition 3 (Satisfiability). $Sat(P)$ denotes that the SMT solver found a model that satisfies P.

Axiom 16 (SMT-Solver Checking). If $\neg Sat(\neg P)$ then $\vdash P$.

Theorem 1 (Quantifier Instantiation Soundness). If $\langle P \rangle$ then $\vdash P$.

Proof. See formalized LEAN proof in `https://github.com/levjj/esverify-theory`. ☐

3 Operational Semantics

Programs are assumed to be in A Normal Form. The dynamic semantics of programs are specified as a small-step evaluation relation over stack configurations to avoid substitution in expressions.

$$
\begin{array}{rcl}
e \in \text{Expressions} & ::= & \text{let } x = \text{true in } e \mid \text{let } x = \text{false in } e \mid \text{let } x = n \text{ in } e \mid \\
& & \text{let } f(x) \text{ req } R \text{ ens } S = e \text{ in } e \mid \text{let } y = \otimes x \text{ in } e \mid \text{let } z = x \oplus y \text{ in } e \mid \\
& & \text{let } y = f(x) \text{ in } e \mid \text{if } (x)\ e \text{ else } e \mid \text{return } x \\
R, S \in \text{Specs} & ::= & t \mid \neg R \mid R \wedge R \mid R \vee R \mid \text{spec } t(x) \text{ req } R \text{ ens } S \\
\kappa \in \text{Configurations} & ::= & (\sigma, e) \mid \kappa \cdot (\sigma, \text{let } y = f(x) \text{ in } e)
\end{array}
$$

$$\boxed{\kappa \hookrightarrow \kappa}$$

$$
\begin{array}{rcllr}
(\sigma, \text{let } y = \text{true in } e) & \hookrightarrow & (\sigma[x \mapsto \text{true}], e) & & [\text{E-TRUE}] \\
(\sigma, \text{let } y = \text{false in } e) & \hookrightarrow & (\sigma[x \mapsto \text{false}], e) & & [\text{E-FALSE}] \\
(\sigma, \text{let } y = n \text{ in } e) & \hookrightarrow & (\sigma[x \mapsto n], e) & & [\text{E-NUM}] \\
(\sigma, \text{let } f(x) \text{ req } R \text{ ens } S = e_1 \text{ in } e_2) & \hookrightarrow & (\sigma[f \mapsto \langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e_1\}, \sigma\rangle], e_2) & & [\text{E-CLOSURE}] \\
(\sigma, \text{let } y = \otimes x \text{ in } e) & \hookrightarrow & (\sigma[y \mapsto v], e) & \text{where } v = \delta(\otimes, \sigma(x)) & [\text{E-UNOP}] \\
(\sigma, \text{let } z = x \oplus y \text{ in } e) & \hookrightarrow & (\sigma[z \mapsto v], e) & \text{where } v = \delta(\oplus, \sigma(x), \sigma(y)) & [\text{E-BINOP}] \\
(\sigma, \text{let } z = f(y) \text{ in } e) & \hookrightarrow & (\sigma_f[g \mapsto \sigma(f), x \mapsto \sigma(y)], e_f) \cdot (\sigma, \text{let } z = f(y) \text{ in } e) & & [\text{E-CALL}] \\
& & \text{where } \sigma(f) = \langle \text{function } g(x) \text{ req } R \text{ ens } S\ \{e_f\}, \sigma_f\rangle & & \\
(\sigma, \text{return } z) \cdot (\sigma_2, \text{let } y = f(x) \text{ in } e_2) & \hookrightarrow & (\sigma_2[y \mapsto \sigma(z)], e_2) & & [\text{E-RETURN}] \\
(\sigma, \text{if } (x)\ e_1 \text{ else } e_2) & \hookrightarrow & (\sigma, e_1) & \text{if } \sigma(x) = \text{true} & [\text{E-IF-T}] \\
(\sigma, \text{if } (x)\ e_1 \text{ else } e_2) & \hookrightarrow & (\sigma, e_2) & \text{if } \sigma(x) = \text{false} & [\text{E-IF-F}] \\
\kappa \cdot (\sigma, \text{let } y = f(x) \text{ in } e) & \hookrightarrow & \kappa' \cdot (\sigma, \text{let } y = f(x) \text{ in } e) & \text{if } \kappa \hookrightarrow \kappa' & [\text{E-CTX}]
\end{array}
$$

$$
\frac{}{\kappa \hookrightarrow^* \kappa} \qquad \frac{\kappa \hookrightarrow^* \kappa' \qquad \kappa' \hookrightarrow \kappa''}{\kappa \hookrightarrow^* \kappa''} \qquad \boxed{\kappa \hookrightarrow^* \kappa}
$$

Definition 4 (Value)**.** If $isValue(\kappa)$ then there exists σ and x such that $\kappa = (\sigma, \text{return } x)$ and $x \in \sigma$.

4 Program Verification

$$
\begin{array}{rcl}
Q[\bullet] \in \text{PropositionContexts} & ::= & P \mid \tau[\bullet] \mid \neg Q[\bullet] \mid Q[\bullet] \wedge Q[\bullet] \mid Q[\bullet] \vee Q[\bullet] \mid pre_1(\otimes, \tau[\bullet]) \mid \\
& & pre_2(\oplus, \tau[\bullet], \tau[\bullet]) \mid pre(\tau[\bullet], \tau[\bullet]) \mid post(\tau[\bullet], \tau[\bullet]) \mid call(\tau[\bullet]) \mid \\
& & \forall x. \{call(x)\} \Rightarrow Q[\bullet] \mid \exists x. Q[\bullet] \\
\tau[\bullet] \in \text{TermContexts} & ::= & \bullet \mid t \mid \otimes \tau[\bullet] \mid \tau[\bullet] \oplus \tau[\bullet] \mid \tau[\bullet](\tau[\bullet])
\end{array}
$$

$$\boxed{P \vdash e : Q}$$

$$\frac{x \notin FV(P) \qquad P \wedge x = \text{true} \vdash e : Q}{P \vdash \text{let } x = \text{true in } e : \exists x.\ x = \text{true} \wedge Q}\ \textsc{vc-tru}$$

$$\frac{x \notin FV(P) \qquad P \wedge x = \text{false} \vdash e : Q}{P \vdash \text{let } x = \text{false in } e : \exists x.\ x = \text{false} \wedge Q}\ \textsc{vc-fls} \qquad \frac{x \notin FV(P) \qquad P \wedge x = n \vdash e : Q}{P \vdash \text{let } x = n \text{ in } e : \exists x.\ x = n \wedge Q}\ \textsc{vc-num}$$

$$\frac{\begin{array}{c} f \notin FV(P) \quad x \notin FV(P) \quad f \neq x \quad x \in FV(R) \\ FV(R) \subseteq FV(P) \cup \{f, x\} \quad FV(S) \subseteq FV(P) \cup \{f, x\} \\ P \wedge \text{spec } f(x) \text{ req } R \text{ ens } S \wedge R \vdash e_1 : Q_1 \quad P \wedge \text{spec } f(x) \text{ req } R \text{ ens } (Q_1[f(x)] \wedge S) \vdash e_2 : Q_2 \\ \langle P \wedge \text{spec } f(x) \text{ req } R \text{ ens } S \wedge R \wedge Q[f(x)] \Rightarrow S \rangle \end{array}}{P \vdash \text{let } f(x) \text{ req } R \text{ ens } S = e_1 \text{ in } e_2 : \exists f.\ \text{spec } f(x) \text{ req } R \text{ ens } (Q_1[f(x)] \wedge S) \wedge Q_2}\ \textsc{vc-func}$$

$$\frac{x \in FV(P) \quad y \notin FV(P) \quad P \wedge y = \otimes x \vdash e : Q \quad \langle P \Rightarrow pre(\otimes, x) \rangle}{P \vdash \text{let } y = \otimes x \text{ in } e : \exists y.\ y = \otimes x \wedge Q}\ \textsc{vc-unop}$$

$$\frac{x \in FV(P) \quad y \in FV(P) \quad z \notin FV(P) \quad P \wedge z = x \oplus y \vdash e : Q \quad \langle P \Rightarrow pre(\oplus, x, y) \rangle}{P \vdash \text{let } z = x \oplus y \text{ in } e : \exists z.\ z = x \oplus y \wedge Q}\ \textsc{vc-binop}$$

$$\frac{\begin{array}{c} f \in FV(P) \quad x \in FV(P) \quad y \notin FV(P) \\ P \wedge call(x) \wedge post(x) \wedge y = f(x) \vdash e : Q \quad \langle P \wedge call(x) \Rightarrow isFunc(f) \wedge pre(f, x) \rangle \end{array}}{P \vdash \text{let } y = f(x) \text{ in } e : \exists y.\ call(x) \wedge post(f, x) \wedge y = f(x) \wedge Q}\ \textsc{vc-app}$$

$$\frac{x \in FV(P) \quad P \wedge x \vdash e_1 : Q_1 \quad P \wedge \neg x \vdash e_2 : Q_2 \quad \langle P \Rightarrow isBool(f) \rangle}{P \vdash \text{if } (x)\ e_1 \text{ else } e_2 : (x \Rightarrow Q_1) \wedge (\neg x \Rightarrow Q_2)}\ \textsc{vc-ite}$$

$$\frac{x \in FV(P)}{P \vdash \text{return } x : x = \bullet}\ \textsc{vc-return}$$

Notation 3 (Function Specs).
spec $t(x)$ req R ens $S \ \equiv\ isFunc(t) \wedge \forall x. \{call(x)\} \Rightarrow ((R \Rightarrow pre(t, x)) \wedge (post(t, x) \Rightarrow S))$

Axiom 17 (L-Function-Application). If $(\sigma[f \mapsto \langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle, x \mapsto v_x], e \longrightarrow^* \sigma', y)$ and $\sigma'(y) = v$ then $\vdash \langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle (v_x) = v$.

Axiom 18 (L-Function-Precondition). Iff $\sigma[f \mapsto \langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle, x \mapsto v_x] \models R$ then $\vdash pre(\langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle, v_x)$.

Axiom 19 (L-Function-Postcondition). If $\vdash \sigma : Q_1$ and $Q_1 \wedge \text{spec } f(x) \text{ req } R \text{ ens } S \wedge R \vdash e : Q_2[\bullet]$ and $\sigma[f \mapsto \langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle, x \mapsto v_x] \models Q_2[f(x)] \wedge S$ then $\vdash post(\langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle, v_x)$.

Axiom 20 (L-Function-Postcondition-Inv). If $\vdash \sigma : Q_1$ and $Q_1 \wedge \text{spec } f(x) \text{ req } R \text{ ens } S \wedge R \vdash e : Q_2[\bullet]$ and $\vdash post(\langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle, v_x)$ then $\sigma[f \mapsto \langle \text{function } f(x) \text{ req } R \text{ ens } S\ \{e\}, \sigma \rangle, x \mapsto v_x] \models Q_2[f(x)] \wedge S$

Theorem 2 (Verification Safety). If $true \vdash e : Q$ and $(\emptyset, e) \hookrightarrow^* \kappa$ then $isValue(\kappa)$ or $\kappa \hookrightarrow \kappa'$ for some κ'.

Proof. See formalized LEAN proof in `https://github.com/levjj/esverify-theory`. □