

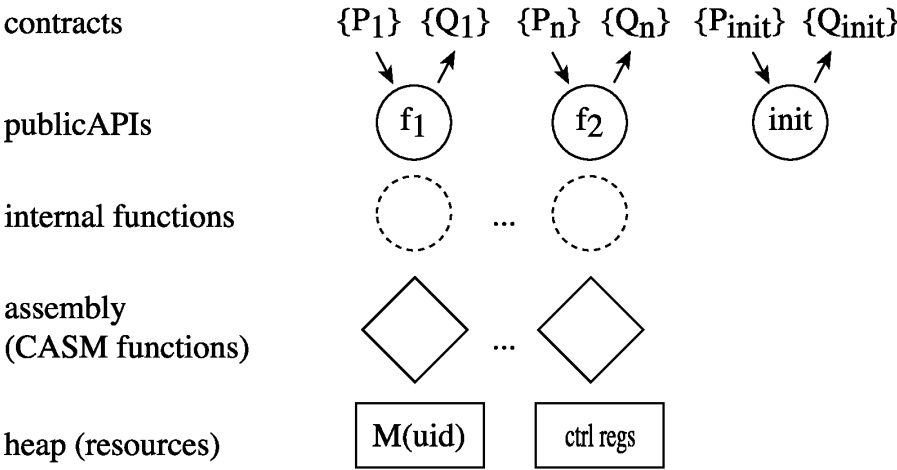


FIG. 1

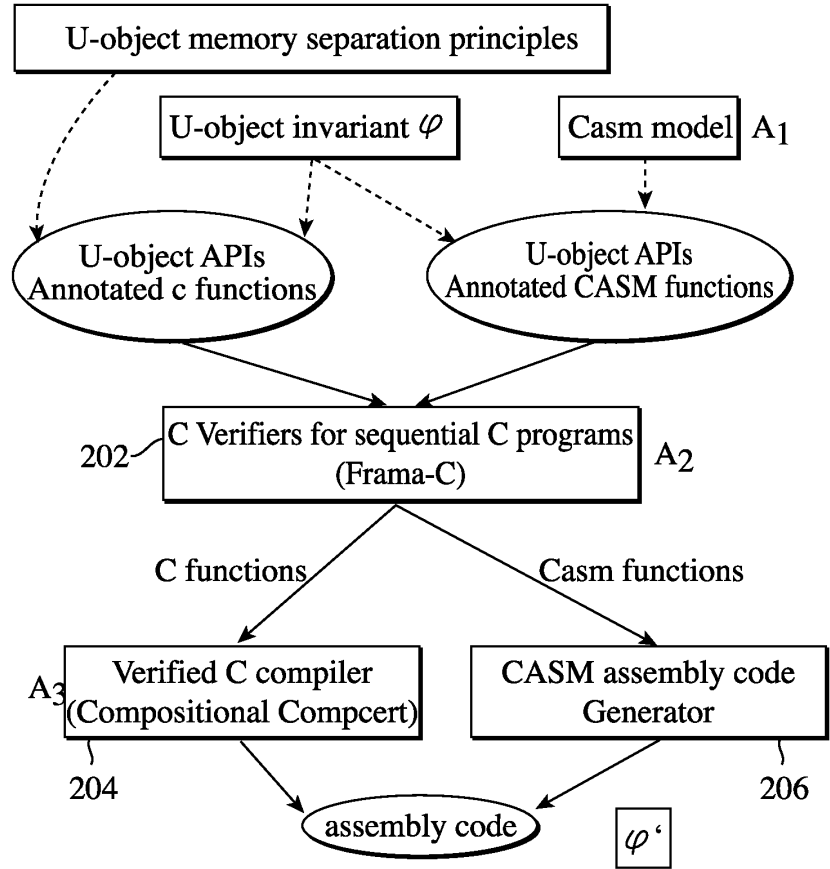


FIG. 2

|                    |        |     |                                                      |
|--------------------|--------|-----|------------------------------------------------------|
| uberobjects        | U      | ::= | · (uid    üobj, U                                    |
| uber object        | üobj   | ::= | (lang, ulock, M, init,                               |
|                    |        |     | →→→→                                                 |
|                    |        |     | pubAPI, casmfd, fd)                                  |
| public API decl    | pubAPI | ::= | $\overrightarrow{f(x:T)} : \tau' = \text{lockUobj};$ |
|                    |        |     | gcmd; unlockUobj                                     |
| CASM fun decl      | casmfd | ::= | $\overrightarrow{f(x:T)} : \tau' = \text{gcasm}$     |
| internal fun decls | fd     | ::= | $\overrightarrow{f(x:T)} : \tau' - \text{gcmd}$      |
| language param     | lang   | ::= | (syntax, →, func)                                    |

FIG. 3

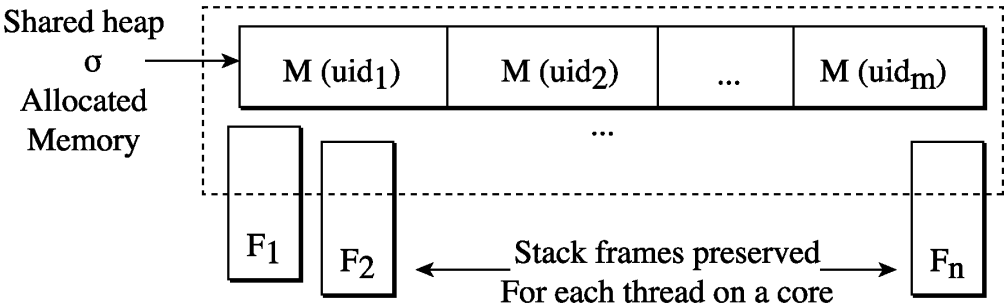


FIG. 4

5/8

CMD

$$\begin{array}{l}
T - \text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p, a), T_1 \quad F \Vdash \text{sl } \sigma, p \xrightarrow{\delta} \sigma', p' \\
T' - \text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p', a), T_1 \quad \text{dom}(\delta') \subseteq \text{uid}. M \cup F \\
\hline
(\mathcal{F}; \sigma; k; (\text{cid}, T), \vec{k}_1; \text{cid}) \xRightarrow{\delta} (\mathcal{F}; \sigma'; (\text{cid}, T'), \vec{k}_1; \text{cid})
\end{array}$$

SWITCH

$$\frac{\text{cid}' \in \text{dom}(\vec{k})}{(\mathcal{F}; \sigma; k; \text{cid}) \xRightarrow{\text{emp}_T} (\mathcal{F}; \sigma; k; \text{cid}')}$$

LOAD

$$\begin{array}{l}
\forall i < n. \text{uid}_i \in \text{dom}(U) \quad \forall i \neq j < n. \text{uid}_i \neq \text{uid}_j \\
F_i \Vdash \text{uid}_i. \text{lang}. \text{initCore}(\text{uid}_i. \text{init}) = p_i \\
\sigma_0 = \text{uid}_i. \text{init}. \text{pre}(a_i) \quad \mathcal{F}_0 ::= F_1 \dots F_n :: \mathcal{F} \\
T_i - \text{tid}_i \mapsto (\text{uid}_i, \text{init}, \text{uid}, \text{lang}, F_i, p_i, a_i) \\
k; - (\text{cid}_1, T_1), \dots, (\text{cid}_n, T_n) \\
\text{cid} \in \text{down}(k) \quad \forall \text{uid} \in U. \text{closed}(\text{uid}. M, \sigma) \\
\hline
\text{uid}_1. \text{init} \parallel \dots \parallel \text{uid}_n. \text{init} \sigma_0 \Rightarrow \text{load} < \mathcal{F}; \sigma; \vec{K}; \text{cid} >
\end{array}$$

UBER-OBJECT-CALL

$$\begin{array}{l}
\mathcal{F} = F_1 :: \dots :: \mathcal{F}' \quad T = \text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p, a), T_1 \\
F \Vdash \text{sl}. \text{extCall}(p) = < \text{uid}' . f \vec{v}, p_1 > \quad \sigma \models \text{uid}. f. \text{pre}(b) \\
F_1 \Vdash p' = \text{uid}' . \text{lang}. \text{initCore}(\text{uid}' . f \vec{v}) \quad \text{fresh}(\text{tid}') \\
\text{uid}' \notin \text{active}(\vec{k}_1) \quad T = \text{tid}' \mapsto (\text{uid}', \text{uid}' . f, \text{uid}' . \text{sl}, F_1, p', b), \\
\text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p_1, a), T_1 \\
\hline
< \mathcal{F}; \sigma; k; < \text{cid}, T >, \vec{k}_1; \text{cid} > \xRightarrow{\text{emp}_{\text{call}}} (\mathcal{F}'; \sigma'; (\text{cid}, T'), \vec{k}_1; \text{cid})
\end{array}$$

FIG. 5A

6/8

$$\begin{array}{c}
\text{UBER-OBJECT-RET} \\
\begin{array}{l}
T = \text{tid}' \mapsto (\text{uid}, \text{mainf}', \text{sl}', F', p', a'), \\
\text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p, a), T_1 \\
F' \Vdash \text{sl}' . \text{halt} (\sigma, p') = \langle v, p_1 \rangle \\
\sigma \Vdash \text{mainf}' . \text{post} (a') \quad F \Vdash \text{sl}' . \text{extRet} (p, v) = p'' \\
T = \text{tid}' \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p'', a), T_1
\end{array} \\
\hline
\langle \mathcal{F}; \sigma; \langle \text{cid}, T \rangle, \vec{k}_1; \text{cid} \rangle \Longrightarrow_{\text{ret}}^{\text{emp}} \langle \mathcal{F}; \sigma'; \langle \text{cid}, T' \rangle, \vec{k}_1; \text{cid} \rangle \\
\text{WAIT} \\
\begin{array}{l}
\vec{k} = \langle \text{cid}, T \rangle, \vec{k}_1 \quad T = \text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p, a), T_1 \\
F \Vdash \text{sl} . \text{extCall}(p) = \langle \text{uid}' . f \vec{v}, p_1 \rangle = \text{uid}' \in \text{active} (k_1)
\end{array} \\
\hline
\langle \mathcal{F}; \sigma; k; \text{cid} \rangle \Longrightarrow_{\text{wait}}^{\text{emp}} \langle \mathcal{F}; \sigma; k; \text{cid} \rangle \\
\text{INTERRUPT} \quad \mathcal{F} = F_1 :: \mathcal{F}' \quad \text{uid}' := \text{uid}_{\text{cidInt}} \\
\begin{array}{l}
F_1 \Vdash \text{uid}' . \text{lang} . \text{initCore} (\text{uid}' . \text{init}) = p' \\
\text{fresh} (\text{tid}') \quad T' = \text{tid}' \mapsto (\text{uid}', \text{uid}' . \text{init}, \text{uid}' . \text{sl}, F_1, p', \_), T \\
\text{uid}' \notin \text{active} (k_1)
\end{array} \\
\hline
\langle \mathcal{F}; \sigma; \langle \text{cid}, T \rangle, \vec{k}_1; \text{cid} \rangle \Longrightarrow_{\text{intStart}}^{\text{emp}} \langle \mathcal{F}'; \sigma; \langle \text{cid}_{\text{int}}, T' \rangle, \vec{k}_1; \text{cid} \rangle \\
\text{INTERRUPT-RET} \\
\begin{array}{l}
\text{uid}' := \text{uid}_{\text{cidInt}} \quad T = \text{tid}' \mapsto (\text{uid}', \text{mainf}', \text{sl}', F', K', \_), \\
T = \text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, K, a), T_1 \\
F' \Vdash \text{sl}' . \text{halt} (\sigma, p') = \langle v, p_1 \rangle \\
\sigma \models \text{mainf}' . \text{post} T' = \text{tid} \mapsto (\text{uid}, \text{mainf}, \text{sl}, F, p, a), T_1
\end{array} \\
\hline
\langle \mathcal{F}; \sigma; k; \langle \text{cid}_{\text{int}}, T \rangle, \vec{k}_1; \text{cid} \rangle \Longrightarrow_{\text{intEnd}}^{\text{emp}} (\mathcal{F}; \sigma'; \langle \text{cid}, T' \rangle, \vec{k}_1; \text{cid}_{\text{int}})
\end{array}$$

FIG. 5B

7/8

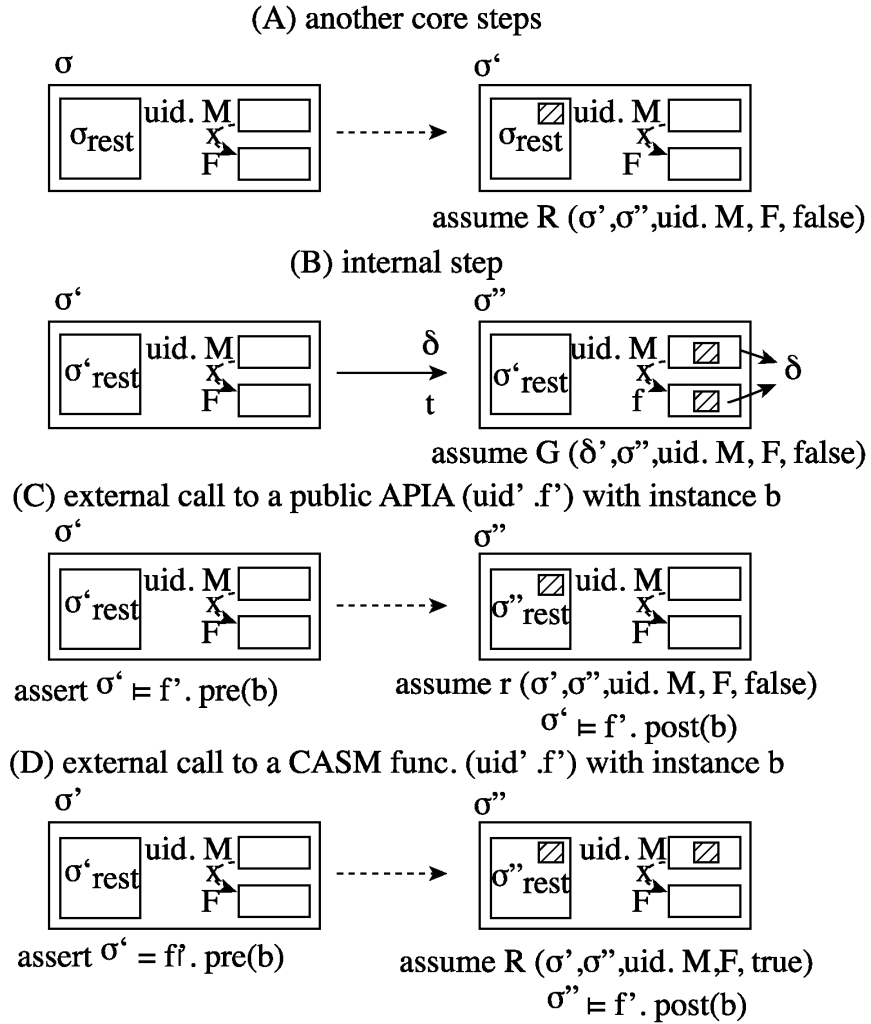


FIG. 6

**#1 RESPECTING THE SPECS****A FUNCTION SATISFIES ITS PRE- AND POST-CONDITIONS**

---

**#2 RESPECTING THE BOUNDARY****[A.] EACH FUNCTION HAS ITS OWN LOCAL STACK.****[B.] NO POINTER TO A STACK LOCATION IS STORED ON THE HEAP.****[C.] NO POINTER IS SENT VIA ARGUMENTS/RETURN VALUES OF AN EXTERNAL CALL.****[D.] A pubAPI OR CASM FUNCTION WRITES/READS FROM ITS STACK AND THE ASSIGNED HEAP.****[E.] NO POINTER TO AN UNALLOCATED LOCATION ON THE HEAP.****FIG. 7**