Gradually Verified Language with Recursive Predicates

Henry Blanchette

Contents

1 Weakest Predonditions

1.1 Concrete Weakest Liberal Precondition Rules

$$\mathsf{WLP} : \textsc{Statement} \times \textsc{SatFormula} \rightarrow \textsc{SatFormula}$$

$$\mathsf{WLP}(s, \phi) := \ \mathsf{match}\ s\ \mathsf{with}$$

$$\left|\begin{array}{lll}
\texttt{skip} & \mapsto & \phi \\
s_1\texttt{;}\ s_2 & \mapsto & \mathsf{WLP}(s_1,\ \mathsf{WLP}(s_2, \phi)) \\
T\ x & \mapsto & \phi \\
x\ \texttt{:=}\ e & \mapsto & \mathsf{footprint}(e)\ \wedge\ [e/x]\phi \\
x\ \texttt{:=}\ \texttt{new}\ C & \mapsto & [\mathsf{newInstance}(C)/x]\phi \\
x.f\ \texttt{:=}\ y & \mapsto & \mathsf{footprint}(x.f)\ \wedge\ [y/x.f]\phi \\
y\ \texttt{:=}\ z.m_C(\overline{e}) & \mapsto & \mathsf{footprint}(\overline{e})\ \wedge\ (z\ \texttt{!=}\ \texttt{null}) \wedge \\
 & & [z/\texttt{this}, \overline{e/x}]\mathsf{pre}(m_C) \wedge \\
 & & \phi - [z/\texttt{this}, \overline{e/\texttt{old}(x)}]\mathsf{post}(m_C) \\
\texttt{if}\ (e)\ \{s_{\text{th}}\}\ \texttt{else}\ \{s_{\text{el}}\} & \mapsto & \mathsf{footprint}(e) \wedge \\
 & & \texttt{if}\ (e)\ \texttt{then}\ (e \wedge \mathsf{WLP}(s_{\text{th}}, \phi)\ \texttt{else}\ (\sim e \wedge \mathsf{WLP}(s_{\text{el}}, \phi)) \\
\texttt{while}\ (e)\ \texttt{invariant}\ \phi_{\text{inv}}\ \{s_{\text{bod}}\} & \mapsto & \mathsf{footprint}(e)\ \wedge\ \phi_{\text{inv}} \wedge \\
 & & \mathsf{WLP}(s_{\text{bod}},\ (\sim e\ \wedge\ \phi) \vee \\
 & & \qquad\qquad \mathsf{WLP}(\texttt{while}\ (e)\ \texttt{invariant}\ \phi_{\text{inv}}\ \{s_{\text{bod}}\},\ \phi)) \\
\texttt{assert}\ \phi_{\text{ass}} & \mapsto & \mathsf{footprint}(\phi_{\text{ass}})\ \wedge\ \phi_{\text{ass}}\ \wedge\ \phi \\
\texttt{hold}\ \phi_{\text{hol}}\ \{s_{\text{bod}}\} & \mapsto & \text{unimplemented} \\
\texttt{release}\ \phi_{\text{rel}} & \mapsto & \text{unimplemented} \\
\texttt{unfold}\ \alpha_C(\overline{e}) & \mapsto & \mathsf{footprint}(\overline{e})\ \wedge\ [\mathsf{unfolded}(\alpha_C(\overline{e}))/\alpha_C(\overline{e})]\phi \wedge \\
 & & [\phi'/\texttt{unfolding}\ \alpha_C(\overline{e})\ \texttt{in}\ \phi'] \\
\texttt{fold}\ \alpha_C(\overline{e}) & \mapsto & \mathsf{footprint}(\overline{e})\ \wedge\ [\alpha_C(\overline{e})/\mathsf{unfolded}(\alpha_C(\overline{e}))]\phi
\end{array}\right.$$

Some utility functions are defined as follows. Note that the implementations of these functions can be made much more efficient than the naive definition here in mathematical notation. For example, calculating the footprint of expressions and formulas can avoid redundancy by not generating permissions that are already satisfied. This can be implemented as implicit in $\wedge$ operations.

$$\mathsf{footprint}(e) := \mathsf{match}\ e\ \mathsf{with}$$

$$\left|\begin{array}{lll} e.f & \mapsto & \mathsf{footprint}(e') \ \wedge\ (e'\ \texttt{!=}\ \texttt{null})\ \wedge\ \texttt{acc}(e'.f) \\ e_1 \oplus e_2 & \mapsto & \mathsf{footprint}(e_1)\ \wedge\ \mathsf{footprint}(e_2) \\ e_1 \otimes e_2 & \mapsto & \mathsf{footprint}(e_1)\ \wedge\ \mathsf{footprint}(e_2) \end{array}\right.$$

$$\mathsf{footprint}(\overline{e}) := \bigwedge \mathsf{footprint}(e)$$

$$\mathsf{footprint}(\alpha_C(\overline{e})) := \alpha_C(\overline{e})\ \wedge\ \mathsf{footprint}(\overline{e})$$

$$\mathsf{footprint}(\phi) := \bigwedge \{\mathsf{footprint}(e) : e \text{ appears in } \phi\}\ \wedge$$

$$\bigwedge \{\mathsf{footprint}(\alpha_C(\overline{e})) : \texttt{unfolding}\ \alpha_C(\overline{e})\ \texttt{in}\ \phi' \text{ appears in } \phi\}$$

$$\mathsf{newInstance}(C) := \text{an object that is a new instance of class } C$$

$$\text{where all fields are assigned to their default values}$$

$$\mathsf{unfolded}(\alpha_C(\overline{e})) := [\overline{e/x}]\mathsf{body}(\alpha_C)$$

$$\left(\bigwedge \phi_i\right) - \left(\bigwedge \psi_j\right) := \bigwedge \{\phi_i : \nexists \psi \subset \{\psi_j\}, \psi \implies \phi_i\}$$

$$\mathsf{pre}(m_C) := \text{the pre-condition of the static contract for } m_C$$

$$\mathsf{post}(m_C) := \text{the post-condition of the static contract for } m_C$$

$$\mathsf{body}(\alpha_C) := \text{the body of predicate } \alpha_C$$