Gradually Verified Language with Recursive Predicates

Henry Blanchette

Contents

1 Weakest Predonditions

1.1 Concrete Weakest Liberal Precondition (WLP) Rules

$$\mathsf{WLP} : \ \textsc{Statement} \times \textsc{FrmSatFormula} \to \textsc{FrmSatFormula}$$

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

$$\left|\begin{array}{lcl}
\texttt{skip} & \mapsto & \phi \\
s_1\texttt{;}\ s_2 & \mapsto & \mathsf{WLP}(s_1,\ \mathsf{WLP}(s_2, \phi)) \\
T\ x & \mapsto & \phi\ \textit{(assert that } x \textit{ does not appear in } \phi\textit{)} \\
x\ \texttt{:=}\ e & \mapsto & \lfloor e \rfloor\ \wedge\ [e/x]\phi \\
x\ \texttt{:=}\ \texttt{new}\ C & \mapsto & [\mathsf{new}(C)/x]\phi \\
x.f\ \texttt{:=}\ y & \mapsto & \lfloor x.f \rfloor\ \wedge\ [y/x.f]\phi \\
y\ \texttt{:=}\ z.m_C(\overline{e}) & \mapsto & \lfloor \overline{e} \rfloor\ \wedge\ z\ \texttt{!=}\ \texttt{null}\ \wedge \\
 & & [z/\texttt{this},\ \overline{e/x}]\mathsf{pre}(m_C)\ * \\
 & & \mathsf{handleMethodPermissions}(m_C, \phi) \\
\texttt{if}\ (e)\ \{s_{\text{the}}\}\ \texttt{else}\ \{s_{\text{els}}\} & \mapsto & \texttt{if}\ (e)\ \texttt{then}\ \mathsf{WLP}(s_{\text{the}}, \phi)\ \texttt{else}\ \mathsf{WLP}(s_{\text{els}}, \phi) \\
\texttt{while}\ (e)\ \texttt{invariant}\ \phi_{\text{inv}}\ \{s_{\text{bod}}\} & \mapsto & \lfloor e \rfloor\ \wedge\ \phi_{\text{inv}}\ \wedge \\
 & & \texttt{if}\ (e)\ \texttt{then}\ \mathsf{WLP}(s_{\text{bod}}, \phi_{\text{inv}})\ \texttt{else}\ \phi \\
\texttt{assert}\ \phi_{\text{ass}} & \mapsto & \lfloor \phi_{\text{ass}} \rfloor\ \wedge\ \phi_{\text{ass}}\ \wedge\ \phi \\
\texttt{hold}\ \phi_{\text{hol}}\ \{s_{\text{bod}}\} & \mapsto & \text{(unimplemented)} \\
\texttt{release}\ \phi_{\text{rel}} & \mapsto & \text{(unimplemented)} \\
\texttt{unfold}\ \alpha_C(\overline{e}) & \mapsto & \lfloor \overline{e} \rfloor\ \wedge\ [\mathsf{unfolded}(\alpha_C(\overline{e}))/\alpha_C(\overline{e}), \\
 & & \ \phi'/\texttt{unfolding}\ \alpha_C(\overline{e})\ \texttt{in}\ \phi']\phi \\
\texttt{fold}\ \alpha_C(\overline{e}) & \mapsto & \lfloor \overline{e} \rfloor\ \wedge\ [\alpha_C(\overline{e})/\mathsf{unfolded}(\alpha_C(\overline{e}))]\phi
\end{array}\right.$$

Since WLP takes a framed, satisfiable formula and yields a framed, satisfiable formula, there is an implicit check that asserts these properties before and after WLP is computed.

1.1.1 Utility Functions

The implementations of the functions in this section 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 permission-subformulas that are already satisfied. This can be implemented as implicit in $\wedge$ by a wrapper $\wedge_{\text{wrap}}$ operation in some way similar to this:

$$\phi \ \wedge_{\text{wrap}} \ \phi' := \begin{cases} \phi & \text{if } \phi \implies \phi' \\ \phi \wedge \phi' & \text{otherwise} \end{cases}$$

The following functions are useful abbreviations for common constructs.

$$\begin{array}{lcl}
\mathsf{new}(C) & := & \text{an object that is a new instance of class } C, \\
 & & \text{where all fields are assigned to their default values} \\
\mathsf{unfolded}(\alpha_C(\overline{e})) & := & [\overline{e/x}]\mathsf{body}(\alpha_C) \\
\mathsf{pre}(z.m_C(\overline{e})) & := & [z/\texttt{this},\ \overline{e/x}]\mathsf{pre}(m_C) \\
\mathsf{pre}(m_C) & := & \text{the static-contract pre-condition of } m_C \\
\mathsf{post}(z.m_C(\overline{e})) & := & [z/\texttt{this},\ \overline{e/\texttt{old}(x)}]\mathsf{post}(m_C) \\
\mathsf{post}(m_C) & := & \text{the static-contract post-condition of } m_C \\
\mathsf{body}(\alpha_C) & := & \text{the body formula of } \alpha_C
\end{array}$$

The footprint function, $\lfloor \cdot \rfloor$, generates a formula containing all the permissions necessary to frame its argument. With efficient implementations of a wrapped $\wedge$, this can result in the smallest such formula.

$$\begin{array}{lcl}
\lfloor e \rfloor & := & \text{match } e \text{ with} \\
 & & \left| \begin{array}{lcl} e.f & \mapsto & \lfloor e' \rfloor \ \wedge \ e' \ \texttt{!= null} \ \wedge \ \texttt{acc}(e'.f) \\ e_1 \oplus e_2 & \mapsto & \lfloor e_1 \rfloor \ \wedge \ \lfloor e_2 \rfloor \\ e_1 \odot e_2 & \mapsto & \lfloor e_1 \rfloor \ \wedge \ \lfloor e_2 \rfloor \\ e & \mapsto & \texttt{true} \end{array} \right. \\
\lfloor \overline{e} \rfloor & := & \bigwedge \lfloor e \rfloor \\
\lfloor \phi \rfloor & := & \bigwedge \{ \lfloor e \rfloor : e \text{ appears in } \phi \} \ \wedge \\
 & & \bigwedge \{ \alpha_C(\overline{e}) : \texttt{unfolding } \alpha_C(\overline{e}) \texttt{ in } \phi' \text{ appears in } \phi \}
\end{array}$$

The $\mathsf{handleMethodPermissions}$ helper function assists in generating the WLP for a method call. It yields a formula that asserts the following:

(a) For all $\texttt{acc}(e.f)$ that appear in $\lfloor \phi \rfloor$, $\texttt{acc}(e.f)$ does not appear in $\mathsf{pre}(z.m_C(\overline{e})) \setminus \mathsf{post}(z.m_C(\overline{e}))$. This asserts that ϕ does not use access to a part of the heap whose access was required to call $z.m_C(\overline{e})$ and then not ensured after the call.

(b) For all $e.f$ that appear in ϕ (not including than those appearing in $\texttt{acc}(e.f)$), $\texttt{acc}(e.f)$ does not appear in $\mathsf{pre}(z.m_C(\overline{e})) \wedge \mathsf{post}(z.m_C(\overline{e}))$. This asserts that ϕ does not rely on the value of a part of the heap that was accessed by $z.m_C(\overline{e})$.

(c) For all $\texttt{unfolding } \alpha_C(\overline{e}) \texttt{ in } \phi'$ that appear in ϕ, $\alpha_C(\overline{e})$ does not appear in $\mathsf{pre}(z.m_C(\overline{e})) \wedge \mathsf{post}(z.m_C(\overline{e}))$. This asserst that ϕ does not rely on the truth of any predicates that were used in $z.m_C(\overline{e})$, as predicates may contain heap accesses.

$$\begin{aligned}
&\mathsf{handleMethodPermissions}(z.m_C(\overline{e}), \phi) := \\
&\qquad \bigwedge \{(\mathsf{pre}(z.m_C(\overline{e})) \setminus \mathsf{post}(z.m_C(\overline{e}))) * \texttt{acc}(e.f) : \lfloor \phi \rfloor \implies \texttt{acc}(e.f)\} \ \land \\
&\qquad \bigwedge \{(\mathsf{pre}(z.m_C(\overline{e})) \land \mathsf{post}(z.m_C(\overline{e}))) * \texttt{acc}(e.f) : e.f \text{ appears in } \phi\} \ \land \\
&\qquad \bigwedge \{(\mathsf{pre}(z.m_C(\overline{e})) \land \mathsf{post}(z.m_C(\overline{e}))) * \alpha_C(\overline{e}) : \texttt{unfolding } \alpha_C(\overline{e}) \texttt{ in} \text{ appears in } \phi\}
\end{aligned}$$

TODO: define without function, $\phi \setminus e$.