

Verifier WLP Definitions

Jenna Wise, Johannes Bader, Jonathan Aldrich, Éric Tanter

November 15, 2018

1 Weakest liberal precondition calculus definitions over self-framed non-gradual formulas

$$\text{WLP}(\text{skip}, \hat{\phi}) = \hat{\phi}$$

$$\text{WLP}(s_1; s_2, \hat{\phi}) = \text{WLP}(s_1, \text{WLP}(s_2, \hat{\phi}))$$

$$\text{WLP}(T \ x := e, \hat{\phi}) = \max_{\Rightarrow} \left\{ \hat{\phi}' \mid \hat{\phi}' \Rightarrow \hat{\phi}[e/x] \quad \wedge \quad \hat{\phi}' \Rightarrow \text{acc}(e) \right\}$$

$$\text{WLP}(\text{if } (x \odot y) \ \{s_1\} \ \text{else } \{s_2\}, \hat{\phi}) =$$

$$\text{WLP}(x.f := y, \hat{\phi}) = \text{acc}(x.f) * \max_{\Rightarrow} \left\{ \hat{\phi}' \mid \hat{\phi}' * \text{acc}(x.f) * (x.f = y) \Rightarrow \hat{\phi} \wedge \hat{\phi}' * \text{acc}(x.f) \in \text{SATFORMULA} \right\}$$

$$\text{WLP}(x := \text{new } C, \hat{\phi}) = \max_{\Rightarrow} \left\{ \hat{\phi}' \mid \hat{\phi}' * (x \neq \text{null}) * \overline{\text{acc}(x.f_i)} \Rightarrow \hat{\phi} \right\}$$

where $\text{fields}(C) = \overline{T_i \ f_i}$

$$\text{WLP}(y := z.m(\bar{x}), \hat{\phi}) = \text{undefined}$$

$$\text{WLP}(y := z.m_C(\bar{x}), \hat{\phi}) = \max_{\Rightarrow} \left\{ \hat{\phi}' \mid y \notin \text{FV}(\hat{\phi}') \quad \wedge \quad \hat{\phi}' \Rightarrow (z \neq \text{null}) * \text{pre}(C, m) \left[z/\text{this}, \overline{x_i/\text{params}(C, m)_i} \right] \right.$$

$$\left. \wedge \quad \hat{\phi}' * \text{post}(C, m) \left[z/\text{this}, \overline{x_i/\text{old}(\text{params}(C, m)_i)}, y/\text{result} \right] \Rightarrow \hat{\phi} \right\}$$

$$\text{WLP}(\text{assert } \phi_a, \hat{\phi}) = \max_{\Rightarrow} \left\{ \hat{\phi}' \mid \hat{\phi}' \Rightarrow \hat{\phi} \quad \wedge \quad \hat{\phi}' \Rightarrow \phi_a \right\}$$

$$\text{WLP}(\text{release } \phi_a, \hat{\phi}) =$$

$$\text{WLP}(\text{hold } \phi_a \ \{s\}, \hat{\phi}) =$$

Note:

Dynamic method calls. Dynamic method calls are left undefined, because we are not verifying programs with dynamic dispatch at this time (all method calls should be static method calls). They are included in the grammar for future implementation.

If & Release & hold. Definitions coming soon.

Predicates in the logic. Although the grammar allows for abstract predicate families, we do not support them yet. Therefore, we assume formulas look like:

$$\phi ::= \text{true} \mid e \odot e \mid \text{acc}(e.f) \mid \phi * \phi$$

2 Algorithmic WLP calculus definitions over self-framed non-gradual formulas

$$\text{WLP}(\text{skip}, \hat{\phi}) = \hat{\phi}$$

$$\text{WLP}(s_1; s_2, \hat{\phi}) = \text{WLP}(s_1, \text{WLP}(s_2, \hat{\phi}))$$

$$\text{WLP}(T \ x := e, \hat{\phi}) = \begin{cases} \hat{\phi}[e/x] & \text{if } \hat{\phi}[e/x] \Rightarrow \text{acc}(e) \\ \text{acc}(e) * \hat{\phi}[e/x] & \text{otherwise} \end{cases}$$

Check that $\text{WLP}(T \ x := e, \hat{\phi}) * x = e \Rightarrow \hat{\phi}$ and that $\text{WLP}(T \ x := e, \hat{\phi})$ is satisfiable.

$$\text{WLP}(\text{if } (x \odot y) \ \{s_1\} \ \text{else } \{s_2\}, \hat{\phi}) =$$

$$\text{WLP}(x.f := y, \hat{\phi}) = \begin{cases} \hat{\phi}[y/x.f] & \text{if } \hat{\phi}[y/x.f] \Rightarrow \text{acc}(x.f) \\ \text{acc}(x.f) * \hat{\phi}[y/x.f] & \text{otherwise} \end{cases}$$

Check that $\text{WLP}(x.f := y, \hat{\phi}) * x.f = y \Rightarrow \hat{\phi}$ and that $\text{WLP}(x.f := y, \hat{\phi})$ is satisfiable.

Important cases to consider:

$$\hat{\phi} = \text{acc}(x.f) * x.f = p * x.f = q * a = b$$

$$\hat{\phi} = \text{acc}(x.f) * \text{acc}(x.f.f) * x = y$$

$$\text{WLP}(x := \text{new } C, \hat{\phi}) = \begin{cases} \hat{\phi} \div x & \text{if } (\hat{\phi} \div x) * x \neq \text{null} * \overline{x \neq e_i} * \overline{\text{acc}(x.f_i)} \Rightarrow \hat{\phi} \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $\text{fields}(C) = \overline{T_i f_i}$, $\hat{\phi} \div x$ means to transitively expand (in-)equalities (\odot) and then remove conjunctive terms containing x , and $\overline{x \neq e_i}$ are conjunctive terms in $\hat{\phi}$.

Check $\text{WLP}(x := \text{new } C, \hat{\phi})$ is satisfiable.

Important cases to consider:

$$\hat{\phi} = x \neq \text{null} * \text{acc}(x.f)$$

$$\hat{\phi} = x \neq \text{null} * \text{acc}(x.f) * x.f = 1 * x.f = y \text{ — should fail, bad postcondition}$$

$$\hat{\phi} = x \neq \text{null} * \text{acc}(x.f) * x = y * x = z \text{ — should fail, bad postcondition}$$

$$\hat{\phi} = x \neq \text{null} * \text{acc}(x.f) * x = y * y = z \text{ — should fail, bad postcondition}$$

$\widehat{\phi} = x \neq \text{null} * \text{acc}(x.f) * x \neq y * y = z$
 $\widehat{\phi} = x \neq \text{null} * \text{acc}(x.f) * \text{acc}(x.f.f) * x.f.f \neq y$ — should fail, bad postcondition
 $\widehat{\phi} = x \neq \text{null} * \text{acc}(y.f) * x = y$ — should fail, bad postcondition
 $\widehat{\phi} = x \neq \text{null} * \text{acc}(x.f) * y > x.f * x.f > z * r \geq x.f * x.f \geq s$ — should fail, bad postcondition

Note:

$x := \text{new } C$ creates a fresh object and assigns it to x without setting default values to the object's fields; therefore, postconditions cannot say anything about the value of x other than it does not equal other values (no aliasing with x) and they cannot say anything about the values of the fields of x .

$$\text{WLP}(y := z.m(\bar{x}), \widehat{\phi}) = \text{undefined}$$

$$\text{WLP}(y := z.m_C(\bar{x}), \widehat{\phi}) =$$

$$\text{WLP}(\text{assert } \phi_a, \widehat{\phi}) =$$

$$\text{WLP}(\text{release } \phi_a, \widehat{\phi}) =$$

$$\text{WLP}(\text{hold } \phi_a \{s\}, \widehat{\phi}) =$$

3 Algorithmic WLP calculus definitions over gradual formulas

$$\widetilde{\text{WLP}}(s, \widehat{\phi}) = \text{WLP}(s, \widehat{\phi}) \text{ where } s \text{ is not a call statement}$$

$$\widetilde{\text{WLP}}(\text{skip}, ? * \phi) = ? * \phi$$

$$\widetilde{\text{WLP}}(s_1; s_2, ? * \phi) = \widetilde{\text{WLP}}(s_1, \widetilde{\text{WLP}}(s_2, ? * \phi))$$

$$\widetilde{\text{WLP}}(T \ x := e, ? * \phi) = ? * \phi[e/x] * e = e$$

Important cases to consider:

$? * \phi = ? * p = q * y.f = 2$ — no mention of e or x for substitution; need extra $* e = e$ because of this case

OR

$$\widetilde{\text{WLP}}(T \ x := e, ? * \phi) = \begin{cases} ? * \phi[e/x] & \text{if } \phi[e/x] \Rightarrow \text{acc}(e) \\ ? * \text{acc}(e) * \phi[e/x] & \text{otherwise} \end{cases}$$

Check that $\widetilde{\text{WLP}}(T \ x := e, ? * \phi) * x = e \Rightarrow ? * \phi$ and that $\widetilde{\text{WLP}}(T \ x := e, ? * \phi)$ is satisfiable.

$$\widetilde{\text{WLP}}(x.f := y, ? * \phi) = \begin{cases} ? * \phi[y/x.f] & \text{if } \phi[y/x.f] \Rightarrow \text{acc}(x.f) \\ ? * \text{acc}(x.f) * \phi[y/x.f] & \text{otherwise} \end{cases}$$

Check that $\widetilde{\text{WLP}}(x.f := y, ? * \phi) * x.f = y \widetilde{\Rightarrow} ? * \phi$ and that $\widetilde{\text{WLP}}(x.f := y, ? * \phi)$ is satisfiable.

Important cases to consider:

$$? * \phi = ? * \text{acc}(x.f) * \text{acc}(x.f.f) * x = y$$

$$\widetilde{\text{WLP}}(x := \text{new } C, ? * \phi) = \begin{cases} ? * \phi \div x & \text{if } (\phi \div x) * x \neq \text{null} * \overline{x \neq e_i} * \overline{\text{acc}(x.f_i)} \Rightarrow \phi \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $\text{fields}(C) = \overline{T_i f_i}$, $\phi \div x$ means to transitively expand (in-)equalities (\odot) and then remove conjunctive terms containing x , and $\overline{x \neq e_i}$ are conjunctive terms in ϕ .

Check that $\widetilde{\text{WLP}}(x := \text{new } C, ? * \phi) * x \neq \text{null} * \overline{x \neq e_i} * \overline{\text{acc}(x.f_i)} \widetilde{\Rightarrow} ? * \phi$ and that $\widetilde{\text{WLP}}(x := \text{new } C, ? * \phi)$ is satisfiable.

Important cases to consider:

Similar to the ones for the non-gradual version; replacing $\hat{\phi}$ with ϕ .

$$\widetilde{\text{WLP}}(\text{assert } \phi_a, ? * \phi) =$$

Note: The static parts of gradual formulas do not need to be self-framed unless the gradual formula is completely precise (completely static). The imprecision can account for the framing.

4 Gradual formula implication and satisfiability

TBD