Verifier WLP Definitions

Jenna Wise, Johannes Bader, Jonathan Aldrich, Éric Tanter

December 7, 2018

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

Important cases to consider:

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

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

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

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

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

Important cases to consider:

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

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

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

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

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

Note:

$x := new\ C$ 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.

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

$\text{WLP}(y := z.m_C(\overline{x}), \widehat{\phi}) = \widehat{\phi}\ \overline{\div\ x_i} \div y \div acc * \text{pre}(C, m)\ \left[z/this, \overline{x_i/\text{params}(C, m)_i}\right]$

where $\widehat{\phi} \div x$ is as defined for the allocation rule, $\widehat{\phi} \div F$ — VERY MUCH IN PROGRESS; NOT CORRECT OR FINISHED

Important cases to consider: What if $y = x_i$ for some argument x_i, z could also be an argument, and $y = z$ $\widehat{\phi} =$

$\text{WLP}(assert\ \phi_a, \widehat{\phi}) = \widehat{\phi}_{acc} * \mid \phi_a \mid * \mid \widehat{\phi} \mid$

where $\mid \phi \mid$ means the formula ϕ without accessibility predicates and $\widehat{\phi}_{acc}$ is the self-framed formula which contains the accessibility predicates that frame $\mid \phi_a \mid * \mid \widehat{\phi} \mid$.

Advice on how to compute $\widehat{\phi}_{acc}$:

1. Transitively expand equalities in $\mid \phi_a \mid * \mid \widehat{\phi} \mid$
2. Determine the list of aliases to each variable or field access using the transitively expanded formula
3. Scan the conjunctive terms in $\mid \phi_a \mid * \mid \widehat{\phi} \mid$ for field accesses
4. For each field access, check if an accessibility predicate exists for it in $\widehat{\phi}_{acc}$ (including if an aliased accessibility predicate exists for it; the list of aliases for each variable should help)
   - If so, continue scanning for field accesses
   - If not, add the accessibility predicate for the field access to $\widehat{\phi}_{acc}$ using the separating conjunction

Important cases to consider:
$\mid \phi_a \mid * \mid \widehat{\phi} \mid = x \neq null * x.f = 1 * x.f = y.f * y.f \neq p * y.f.f > 1$
$\mid \phi_a \mid * \mid \widehat{\phi} \mid = x = y * x = z * x.f = 8 * y.f > 4$
$\mid \phi_a \mid * \mid \widehat{\phi} \mid = x = y * y = z * z.f + 1 \leq 10 * \text{true}$
$\mid \phi_a \mid * \mid \widehat{\phi} \mid = x.f.f \neq y * x \neq p * p = q * x.f > y * y > z$
$\mid \phi_a \mid * \mid \widehat{\phi} \mid = y > x.f * x.f > z * r \geq x.f * x.f \geq s$

Assumptions:

We assume that objects are only referred to in formulas through variables or as field accesses, variables or field accesses which refer to objects are not used in binary operations ($\oplus$), and variables or field accesses which refer to objects are not used in comparison operators other than $\neq$ and $=$.

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

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

3 Algorithmic WLP calculus definitions over gradual formulas

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

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

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

$\widetilde{\text{WLP}}(T\ x := e, ? * \phi) = ? * \phi[e/x] * e = e$
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
$$\widetilde{\text{WLP}}(T\ x := e, ? * \phi) = \begin{cases} ? * \phi[e/x] & \mathit{if}\ \phi[e/x] \Rightarrow \mathit{acc}(e) \\ ? * \mathit{acc}(e) * \phi[e/x] & \mathit{otherwise} \end{cases}$$
Check that $\widetilde{\text{WLP}}(T\ x := e, ? * \phi) * x = e \mathrel{\widetilde{\Rightarrow}} ? * \phi$ and that $\widetilde{\text{WLP}}(T\ x := e, ? * \phi)$ is satisfiable.

$$\widetilde{\text{WLP}}(x.f := y, ? * \phi) = \begin{cases} ? * \phi[y/x.f] & \mathit{if}\ \phi[y/x.f] \Rightarrow \mathit{acc}(x.f) \\ ? * \mathit{acc}(x.f) * \phi[y/x.f] & \mathit{otherwise} \end{cases}$$
Check that $\widetilde{\text{WLP}}(x.f := y, ? * \phi) * x.f = y \mathrel{\widetilde{\Rightarrow}} ? * \phi$ and that $\widetilde{\text{WLP}}(x.f := y, ? * \phi)$ is satisfiable.
Important cases to consider:
$? * \phi = ? * \mathit{acc}(x.f) * \mathit{acc}(x.f.f) * x = y$

$$\widetilde{\text{WLP}}(x := \mathit{new}\ C, ? * \phi) = \begin{cases} ? * \phi \div x & \mathit{if}\ (\phi \div x) * x \neq \mathit{null} * \overline{x \neq e_i} * \overline{\mathit{acc}(x.f_i)} \Rightarrow \phi \\ \mathit{undefined} & \mathit{otherwise} \end{cases}$$
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 $\widetilde{\text{WLP}}(x := \mathit{new}\ C, ? * \phi) * x \neq \mathit{null} * \overline{x \neq e_i} * \overline{\mathit{acc}(x.f_i)} \mathrel{\widetilde{\Rightarrow}} ? * \phi$ and that $\widetilde{\text{WLP}}(x := \mathit{new}\ C, ? * \phi)$ is satisfiable.
Important cases to consider:
Similar to the ones for the non-gradual version; replacing $\widehat{\phi}$ with ϕ.

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

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