SPECIFICATIONS & REFINEMENTS

Alessandro Coglio

Kestrel Institute

General Concepts

S set of (all possible) specifications

$\rightsquigarrow \subseteq S \times S$ refinement relation

$$\left.\begin{array}{ll} \boxed{\text{REFL}}\ s \rightsquigarrow s & \text{— reflexive} \\ \boxed{\text{TRANS}}\ s \rightsquigarrow s' \wedge s' \rightsquigarrow s'' \Rightarrow s \rightsquigarrow s'' & \text{— transitive} \end{array}\right\} \text{ preorder relation}$$

$\rightsquigarrow^+$ transitive closure of $\rightsquigarrow$

$\rightsquigarrow^*$ reflexive and transitive closure of $\rightsquigarrow$

P set of (all possible) programs in some target programming language

$S \cap P \neq \emptyset$ if the specification language is a superset of a subset of the programming language

$C : S \xrightarrow{p} P$ code generation (partial function, indicated by $\xrightarrow{p}$, i.e. domain $\mathcal{D}(C) \subseteq S$)

$\mathcal{D}(C) = S \cap P \wedge C = id$ if $S \cap P \neq 0$ (case above)

$s_0 \rightsquigarrow s_1 \rightsquigarrow \dots \rightsquigarrow s_n \overset{C}{\longmapsto} p$ derivation of p from requirements s_0 via intermediate $s_1, \dots, s_n$

$I : S \to 2^P$ possible implementations of specifications

$I(s) \triangleq \{ p \in P \mid \exists s' \in \mathcal{D}(C) .\ s \rightsquigarrow^* s' \wedge p = C(s') \} \subseteq P$

$\boxed{\text{MONO}} \vdash s \rightsquigarrow s' \Rightarrow I(s) \supseteq I(s')$

$$\begin{tikzcd}
p \in I(s') \overset{I}{\Rightarrow} \exists s'' \in \mathcal{D}(C) . & s' \rightsquigarrow^* s'' \wedge \arrow[dd] & \\
s \rightsquigarrow s' \arrow[dr, "\text{TRANS}"] & & p = C(s'') \arrow[d] \\
& s \rightsquigarrow^* s'' \arrow[r] & p \in I(s)
\end{tikzcd}$$

QED

$\boxed{\text{CIMP}} \vdash s \in \mathcal{D}(C) \Rightarrow C(s) \in I(s)$

$$\begin{tikzcd}
\text{REFL} \arrow[r] & s \rightsquigarrow^* s \arrow[dr] & \\
& & C(s) \in I(s) \\
& C(s) = C(s) \arrow[ur] &
\end{tikzcd}$$

QED

$$\begin{tikzcd}
s_0 \arrow[rr, rightsquigarrow] \arrow[d, "I"] & & s_1 \arrow[rr, rightsquigarrow] \arrow[d, "I"] & & \cdots \arrow[rr, rightsquigarrow] & & s_n \arrow[rr, mapsto, "C"] \arrow[d, "I"] & & p \arrow[d, "\{\cdot\}"] \\
I(s_0) & \supseteq & I(s_1) & \supseteq & \cdots & \supseteq & I(s_n) & \supseteq & \{p\} \\
& & & \text{MONO} \arrow[ull] \arrow[u] \arrow[urr] & & & & \text{CIMP} \arrow[u] &
\end{tikzcd}$$

derivation (see above)

monotonically decreasing sequence of sets of programs, ending in singleton

Pop-Refinement

idea: $S \triangleq 2^P$, $\leadsto \triangleq \supseteq$, $\mathcal{C} \triangleq \{\langle \tilde{p}, p \rangle \in 2^P \times P \mid \tilde{p} = \{p\}\}$

$\Rightarrow I = id$ — no indirection : the specification can constrain every aspect of the target program

more precisely, in a general-purpose logical language (especially, in a theorem prover):

P consists of deeply embedded target programs

S consists of predicates over P

$\leadsto$ is (backward) logical implication

$s_0(p) \triangleq \ldots$ — requirements (functional, non-functional, program-level, syntactic, etc.)

$s_i(p) \Rightarrow s_{i-1}(p)$ — refinement step

$s_n(p) \triangleq [p = \hat{p}]$ — p_0 is a program in explicit syntactic form — the implementation

$\mathcal{C}$ trivially turns $[p = \hat{p}]$ into $\hat{p}$ — without even pretty-printing if P consist of concrete syntax

$s_0(p) \Leftarrow s_1(p) \Leftarrow \ldots \Leftarrow s_n(p) = [p = \hat{p}] \xmapsto{\mathcal{C}} \hat{p}$ — derivation — all expressed in the logic

$\vdash s_0(\hat{p})$ follows from the derivation

in ACL2, this can all fit in its first-order logic:

- formalize syntax and semantics of the target programming language
- P is a set of ACL2 values
- each s_i is an ACL2 function (predicate) over P
- the final implementation p_0 is an ACL2 value

Shallow Pop-Refinement

idea: functions in the logic can be regarded as shallowly embedded programs, given code generator C
— sufficient to explicitly specify and refine functional constraints (only)

more precisely, in ACL2:

$f_1 : \mathcal{U}^{n_1} \to \mathcal{U}^{m_1}, \ldots, f_p : \mathcal{U}^{n_p} \to \mathcal{U}^{m_p}$ target functions ($p \geq 1$)

$S \subseteq (\mathcal{U}^{n_1} \to \mathcal{U}^{m_1}) \times \cdots \times (\mathcal{U}^{n_p} \to \mathcal{U}^{m_p})$ specification (2nd-order — needs SOFT or apply$)

$S_0(f_1, \ldots, f_p) \triangleq \ldots$ — requirements (functional only, but including hyperproperties)

$S_i(f_1, \ldots, f_p) \Rightarrow S_{i-1}(f_1, \ldots, f_p, f_{p+1}, \ldots)$ — refinement step (may add function variables)

$S_n(f_1, \ldots, f_p, \ldots) \triangleq [f_1 = \hat{f}_1] \wedge \cdots \wedge [f_p \triangleq \hat{f}_p] \wedge \cdots$ — each $\hat{f}_i \triangleq \ldots$ is a defined function

$$\text{implementation} \begin{cases} \{\hat{f}_1, \ldots, \hat{f}_p, \ldots\} & \text{— in (executable) ACL2} \\ C(\{\hat{f}_1, \ldots, \hat{f}_p, \ldots\}) = \ldots & \text{— in some other programming language} \end{cases}$$

notation: above we use uppercase S instead of lowercase s for specification predicates, which is unambiguous also because in the context of shallow pop-refinement in ACL2 we do not explicitly reference the set of all possible such specification predicates

Some Shallow Pop-Refinement Specification Forms

$\boxed{\text{PP}}$ $S(f) \triangleq [\forall x.\ \Phi(x) \Rightarrow \Psi(x, f(x))]$

Φ precondition , Ψ postcondition
generalizes from $f : \mathcal{U} \to \mathcal{U}$ to $f : \mathcal{U}^n \to \mathcal{U}^m$

$\boxed{\text{Rf}}$ $S(f) \triangleq [\forall x.\ R(x, f(x))]$

R input/output relation
generalizes from $f : \mathcal{U} \to \mathcal{U}$ to $f : \mathcal{U}^n \to \mathcal{U}^m$

PP is special case of Rf: $R(x,y) := [\Phi(x) \Rightarrow \Psi(x,y)]$

$\boxed{\text{Rf}\alpha}$ $S(f) \triangleq [\forall x, \bar{x}.\ R(x, \bar{x}, f(x, \bar{\alpha}(\bar{x})))]$

x selected input — e.g. targeted by a transformation (not necessarily the first in $x, x_1, \ldots, x_n$)
$\bar{x} = x_1, \ldots, x_n$ additional inputs , $n \geqslant 0$
$\alpha_1, \ldots, \alpha_p : \mathcal{U}^n \to \mathcal{U}$ argument functions
$R \subseteq \mathcal{U}^{1+n} \times \mathcal{U}^m$ input/output relation , on $f \circ \bar{\alpha}$

Rf is special case of Rfα : $p := n$, $\alpha_1 := id$, … , $\alpha_p := id$

more forms may be added here as needed