Verifier Core Language BNF Grammar

Jenna Wise, Johannes Bader, Henry Blanchette, Jonathan Aldrich, Éric Tanter

June 7, 2019

x, y, z	$\in$	*VAR*	*(variables)*
v	$\in$	*VAL*	*(values)*
e	$\in$	*EXPR*	*(expressions)*
s	$\in$	*STMT*	*(statements)*
o	$\in$	*LOC*	*(object Ids)*
f	$\in$	*FIELDNAME*	*(field names)*
m	$\in$	*METHODNAME*	*(method names)*
C, D	$\in$	*CLASSNAME*	*(class names)*
α	$\in$	*PREDNAME*	*(predicate names)*
P	$::=$	$\overline{cls}\ s$	
cls	$::=$	$\texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{field}\ \overline{pred}\ \ \overline{method}\}$	
field	$::=$	$T\ f;$	
pred	$::=$	$\texttt{predicate}\ \alpha_C(\overline{T\ x}) = \widetilde{\phi}$	
T	$::=$	$\texttt{int} \mid \texttt{bool} \mid C \mid \top$	
method	$::=$	$T\ m(\overline{T\ x})\ \texttt{dynamically}\ contract\ \texttt{statically}\ contract\ \{s\}$	
contract	$::=$	$\texttt{requires}\ \widetilde{\phi}\ \texttt{ensures}\ \widetilde{\phi}$	
$\oplus$	$::=$	$+ \mid - \mid * \mid \backslash \mid \&\& \mid \mid\mid$	
$\odot$	$::=$	$\neq \mid = \mid < \mid > \mid \leq \mid \geq$	
s	$::=$	$\texttt{skip} \mid s_1\ ;\ s_2 \mid T\ x \mid x := e \mid \texttt{if}\ (e)\ \{s_1\}\ \texttt{else}\ \{s_2\} \mid \texttt{while}\ (e)\ \texttt{invariant}\ \widetilde{\phi}\ \{s\} \mid x.f := y$	
		$\mid x := \texttt{new}\ C \mid y := z.m(\overline{x}) \mid y := z.m_C(\overline{x}) \mid \texttt{assert}\ \phi \mid \texttt{release}\ \phi \mid \texttt{hold}\ \phi\ \{s\} \mid \texttt{fold}\ A$	
		$\mid \texttt{unfold}\ A$	
e	$::=$	$v \mid x \mid e \oplus e \mid e \odot e \mid e.f$	
x	$::=$	$\texttt{result} \mid id \mid \texttt{old}(id) \mid \texttt{this}$	
v	$::=$	$n \mid o \mid \texttt{null} \mid \texttt{true} \mid \texttt{false}$	
A	$::=$	$\alpha(\overline{e}) \mid \alpha_C(\overline{e})$	
$\circledast$	$::=$	$\wedge \mid *$	
ϕ	$::=$	$e \mid A \mid \texttt{acc}(e.f) \mid \phi \circledast \phi \mid (\texttt{if}\ e\ \texttt{then}\ \phi\ \texttt{else}\ \phi) \mid (\texttt{unfolding}\ A\ \texttt{in}\ \phi)$	
$\widetilde{\phi}$	$::=$	$\phi \mid ? * \phi$	