

Verifier Core Language BNF Grammar

Jenna Wise, Johannes Bader, Jonathan Aldrich, Éric Tanter

May 30, 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)
A	$\in PREDNAME$	(predicate names)
P	$::= \overline{cls} \ s$	
cls	$::= \text{class } C \text{ extends } D \ \{\overline{field} \ \overline{pred} \ \overline{method}\}$	
$field$	$::= T \ f;$	
$pred$	$::= \text{predicate } A_C(\overline{T} \ x) = \tilde{\phi}$	
T	$::= \text{int} \mid C \mid \top$	
$method$	$::= T \ m(\overline{T} \ x) \ \text{dynamic contract } \{s\} \mid T \ m(\overline{T} \ x) \ \text{static contract } \{s\}$	
$contract$	$::= \text{requires } \tilde{\phi} \ \text{ensures } \tilde{\phi}$	
\oplus	$::= + \mid - \mid * \mid \backslash$	
\odot	$::= \neq \mid = \mid < \mid > \mid \leq \mid \geq$	
s	$::= \text{skip} \mid s_1 ; s_2 \mid T \ x \mid x := e \mid \text{if } (e) \{s_1\} \text{ else } \{s_2\} \mid \text{while } (e) \text{ inv } \tilde{\phi} \{s\}$ $\mid x.f := y \mid x := \text{new } C \mid y := z.m_C(\overline{x}) \mid \text{assert } \phi \mid \text{release } \phi \mid \text{hold } \phi \{s\}$ $\mid \text{fold } A(\overline{e}) \mid \text{unfold } A(\overline{e})$	
e	$::= v \mid x \mid e \oplus e \mid e \odot e \mid e.f$	
x	$::= \text{result} \mid id \mid \text{old}(id) \mid \text{this}$	
v	$::= n \mid o \mid \text{null} \mid \text{true} \mid \text{false}$	
ϕ	$::= e \mid A(\overline{e}) \mid \text{acc}(e.f) \mid \phi \wedge \phi \mid \phi * \phi \mid (\text{if } e \text{ then } \phi \text{ else } \phi)$	
$\tilde{\phi}$	$::= \phi \mid ? * \phi$	