Translational Semantics of ASM language in Boogie

Zheng Cheng, Rosemary Monahan, and James F. Power

Computer Science Department, Maynooth University, Co. Kildare, Ireland
{zcheng, rosemary, jpower}@cs.nuim.ie

ASM Instruction (S)	**Corresponding Boogie Statements ([[S]])**
	Stack Handling Instructions
push$_\tau$ c	Stk := $[\![c]\!]$::Stk ; (where c is a constant of type $\tau \in \{int, bool, string\}$)
pop	assert size(Stk)>0 ; Stk := tl(Stk) ;
store x	assert size(Stk)>0 ; x := hd(Stk) ; Stk := tl(Stk) ; (where x is a variable)
load x	Stk := x::Stk ; (where x is a variable)
swap	assert size(Stk)>1 ; Stk := hd(tl(Stk))::hd(Stk)::tl(tl(Stk)) ;
dup	assert size(Stk)>0 ; Stk := hd(Stk)::Stk ;
dup_x1	assert size(Stk)>1 ; Stk := hd(Stk)::hd(tl(Stk))::hd(Stk)::tl(tl(Stk)) ;
	Control Instructions
if $Stmt_1$ **else** $Stmt_2$	var cond$^\#$: bool ; assert size(Stk) > 0 ; cond$^\#$:= hd(Stk) ; Stk := tl(Stk) ; if (cond$^\#$) $[\![Stmt_1]\!]$ else $[\![Stmt_2]\!]$
goto n	goto l ; (where l is a fresh label. It lables the program point which corresponds to the ASM instruction offset n)
iter $Stmt_1$ **enditer**	var col$^\#$: Collection ; assert size(Stk) > 0 ; col$^\#$:= hd(Stk) ; Stk := tl(Stk) ; while (hasNext(col$^\#$)) INV { Stk := next(col$^\#$)::Stk ; $[\![Stmt_1]\!]$}
pcall sig	$let\ n = arg\_size(sig)\ in$ $let\ \overline{args} = tk(Stk, n),\ ctx = hd(dp(Stk, n))\ in$ assert size(Stk) > n ; call invoke(reflect(sig, ctx), $\overline{args}$) ; Stk := dp(Stk, n+1) ;
call sig	$let\ n = arg\_size(sig)\ in$ $let\ \overline{args} = tk(Stk, n),\ ctx = hd(dp(Stk, n))\ in$ var result$^\#$: T ; assert size(Stk) > n ; call result$^\#$:= invoke(reflect(sig, ctx), $\overline{args}$) ; Stk := result$^\#$::dp(Stk, n+1) ; (where T is the return type of the reflected method)
	Model Handling Instructions
new r	$let\ mm = hd(Stk),\ cl = hd(tl(Stk))\ in$ $let\ clazz = resolve(mm, cl)\ in$: var $r^\#$: Ref ; havoc $r^\#$; assume $r^\# \neq null \wedge \neg$read(heap, $r^\#$, $alloc$) ; assert size(Stk) > 1 ; assume typeof($r^\#$) = $clazz$; heap := update(heap,$r^\#$,$alloc$,$true$); Stk := $r^\#$::tl(tl(Stk)) ;
get f	$let\ o = hd(Stk)\ in$ assert size(Stk) > 0 $\wedge$ $o \neq null \wedge$ read(heap, o, $alloc$) ; Stk := read(heap,o,f)::tl(Stk) ;
set f	$let\ o = hd(tl(Stk)),\ v = hd(Stk)\ in$ assert size(Stk) > 1 $\wedge$ $o \neq null \wedge$ read(heap, o, $alloc$) ; if (isCollection(f)) { heap := update(heap,read(heap,o,f),read(heap,o,f) $\cup$ v); } else { heap := update(heap,o,f,v) ; } Stk := tl(tl(Stk));
findme	$let\ mm = hd(Stk),\ cl = hd(tl(Stk))\ in$ assert size(Stk) > 1 ; Stk := resolve(mm, cl)::tl(tl(Stk)) ;
getasm	Stk := ASM::Stk ;