PCC

Martin Wildmoser

7th February 2005

Contents

theory *EX-MainSumAuto = VCGExec*:

0.0.1 Definitions

0.0.2 Name Declarations

constdefs
this::*nat*
this ≡ *0*
k::*nat*
k≡*2*
n::*nat*
n ≡ *1*
res::*nat*
res≡*3*

0.0.3 Program Code

consts *comment* ::*instr* ⇒ *string* ⇒ *instr* ((- −− -) [*61*,*60*] *60*)
defs *comment-def* [*simp*]:
comment i s ≡ *i*

constdefs *sum*::*cname*
sum ≡ *''sum''*

constdefs
StartCR :: *jvm-method cdecl*
StartCR ≡ (*Start*,(*Object*, [],[(*main*,[],*Integer*,(*1*,*1*,[*Push* (*Intg 5*),*Return*],[])),
(*sum*,[*Integer*],*Integer*,(*2*,*4*,[
Push (*Intg 0*),
Store res,
Push (*Intg 0*),
Store k,
Load n,
Push (*Intg 65536*),
IfIntLeq 16,
Load k −−*''LOOP* : *sum-inv ''*,
Load n,
IfIntLeq 13 −− *''if n* <= *k then goto EXIT else k*++; *res*+=*k*; *goto LOOP''* ,
Load res,
Push (*Intg 2147418113*) −− *''maximum value Rg res can have at this position without causing an overflow''*,
IfIntLeq 10,
Load k ,
Push (*Intg 1*),
IAdd,
Store k,

Load k,
Load res,
IAdd,
Store res,
Goto (−14) −− ''goto LOOP'',
Load res −− ''EXIT'',
Return],
[])])])

constdefs
prog::*jbc-prog*
prog ≡ (*SystemClasses* @ [*StartCR*],[])

0.0.4 Generating ML Code

generate-code (*EX-MainSumAuto.ML*) [*term-of*]
exprg = *exprg*
phi-exprg = *map-of2* (*convert-pt* (*prog-kil exprg*))
wf-jvm-prog-phi = λ Φ (*P*::*jvm-prog*). *wf-jvm-prog-phi* Φ *P*
wf = *wf*
opt = *opt*
vcg = *vcgTy*
prog = *prog*
phi-prog = *map-of2* (*convert-pt* (*prog-kil* (*fst prog*)))
initjob = *initjob* (*fst prog*) *Start ''sum''*
nxtjob = *nxtjob* (*fst prog*) *Start ''sum''*
printjob = λ *job*. *printjob* (*fst prog*) *Start ''sum''* *job*
parsejob = *parsejob*

0.0.5 Generate annotations

ML {∗ *print-depth 100* ∗}
ML {∗ *use pa-sources* ∗}
ML {∗ *use-src /home/wiss/wildmosm/work/PA/JVM/ EX-MainSumAuto*; ∗}
ML {∗ *val aprog* = *interval-analyzeprog prog*; ∗}
ML {∗ *aprog*; ∗}

0.0.6 Computing and verifying the verification condition

ML {∗ *wf aprog*; ∗}
ML {∗ *val vc* = *opt* (*vcg aprog*); ∗}

— now we trasfer the ML result back to Isabelle
ML-setup {∗
val t = *term-of-expr vc*;
val T = *fastype-of t*;
Context.>> (*fn thy => thy |>*

Theory.add-consts-i [(*vc*, *T*, *NoSyn*)] |>
(*fst o PureThy.add-defs-i false* [((*vc-def*, *Logic.mk-equals* (*Const* (*EX-MainSumAuto.vc*, *T*), *t*)), []
)]));
∗}

— the verification condition is now defined as constant vc::expr

lemma *vc-prg-holds*:
prog ⊢ *vc*
apply (*simp only*: *provable-def vc-def*)
apply (*safe intro*!: *And0 AndI'*)
apply (*simp only*: *deriv-def*, *rule ballI*, *clarsimp simp del*: *evalE-evalEs.simps simp add*: *sem-simps*)
done

end