theory *SALOverflowPlatform* = *SALSafetyLogic* + *TupleOrd*:

1 SAL Overflow Platform

1.1 Wellformedness Checker

consts
checkPos :: *SALprogram* ⇒ (*pos list*) ⇒ *bool*

primrec
checkPos prg [] = *True*
checkPos prg (*pc* # *pcs*) = (*if* (*let* (*pn*, *i*) = *pc in* (*case* (*cmd prg pc*) *of*
None ⇒ *False*
| *Some c* ⇒ (*case c of*
SET x n ⇒ *True*
| *ADD x y* ⇒ *True*
| *SUB x y* ⇒ *True*
| *INC x* ⇒ *True*
| *JMPEQ x y t* ⇒ (*t* = *0*) ⟶ ((*anF prg* (*pn*, *i*)) ≠ *None*)
| *JMPL x y t* ⇒ (*t* = *0*) ⟶ ((*anF prg* (*pn*, *i*)) ≠ *None*)
| *JLE x y t* ⇒ (*t* = *0*) ⟶ ((*anF prg* (*pn*, *i*)) ≠ *None*)
| *JMPB t* ⇒ ((*anF prg* (*pn*, *i* − *t*)) ≠ *None*)
| *CALL x pn′* ⇒ (*anF prg* (*pn′*, *0*) ≠ *None*)
| *RET x* ⇒ (*list-all* (λ (*pc*, *B*). *anF prg pc* ≠ *None*) (*ret-succs prg pc x* (*callpoints prg pn*))) ∧ (*0* < *pn*)
| *MOV s t* ⇒ *True*
| *HALT* ⇒ *True*
)))
then (*checkPos prg pcs*)
else False)

constdefs
wf :: *SALprogram* ⇒ *bool*
wf prg ≡ *checkPos prg* (*domC prg*)

1.2 Environment Functions

constdefs
callstate :: *env* ⇒ *SALstate*
callstate ≡ λ *e*. (*case* (*cs e*) *of*
[] ⇒ *arbitrary*
| (*k*, *m′*) # *css* ⇒ ((*h e*)!*k*, *m′*, *e*(|*cs* := *css*, *h* := *take k* (*h e*)|)))

constdefs
callmem :: *env* ⇒ *tram*
callmem e ≡ *snd* (*hd* (*cs e*))

constdefs
callpc :: *env* ⇒ *pos*

callpc e ≡ (*h e*)!(*fst* (*hd* (*cs e*)))

constdefs
callenv :: *env* ⇒ *env*
callenv e ≡ (*let* (*k*,*m*)=*hd* (*cs e*) *in e*(| *cs*:= *tl* (*cs e*), *h*:= (*take k* (*h e*))|))

syntax
callmem :: *env* ⇒ *tram* (‾*m* -)
callpc :: *env* ⇒ *pos* (‾*pc* -)
callenv :: *env* ⇒ *env* (‾*e* -)

1.3 Instantiating the VCG

constdefs
domA :: *SALprogram* ⇒ *pos list*
domA ≡ λ *prg*. [*pc*∈ *domC prg*. (*anF prg pc*) ≠ *None*]

constdefs
incA :: *pos* ⇒ *pos*
incA ≡ λ (*pn*,*i*). (*pn*,*Suc i*)

constdefs
vcgSAL :: *SALprogram* ⇒ *SALform*
vcgSAL prg ≡ *vcG Conj Impl FalseF ipc initF safeF succsF wpF domC domA anF prg*

1.4 Generating ML Code for the VCG

types-code
set ((- −> *bool*))

consts-code
op : ((- |> -))
Collect ((-))

ML {∗ *fun term-of-fun-type - - - -= error term-of-fun-type applied* ∗}

ML {∗ *fun term-of-bool - = error term-of-bool applied* ∗}

generate-code (*vcgSAL.ML*) [*term-of*]
prov = *provable*
vcg = *vcgSAL*

— You find the executable VCG in vcgOSAL.ML. Instructions on how to use it are in README.txt

end