theory *SALTypeSafetyPlatform* = *VerificationConditionGenerator* + *SALSemantics*:

1 SAL Safety Logic

constdefs
valid::*SALprogram* ⇒ *SALstate* ⇒ *SALform* ⇒ *bool* ((-,- ⊨ -) [*61*,*61*,*60*] *60*)
valid prg s f ≡ *f*(*s*)

constdefs
provable::*SALprogram* ⇒ *SALform* ⇒ *bool* ((- ⊢ -) [*61*,*60*] *60*)
provable prg f ≡ ∀ *s*. *valid prg s f*

lemma [*code*]:
provable prg f = *term* (*provable* (*to-term prg*) (*to-term f*))
apply (*simp add*: *term-def to-term-def*)
done

constdefs
FalseF::*SALform*
FalseF ≡ λ *s*. *False*

lemma [*code*]:
FalseF = *term* (λ *s*. *False*)
apply (*simp only*: *term-def FalseF-def*)
done

constdefs
TrueF::*SALform*
TrueF ≡ λ *s*. *True*

lemma [*code*]:
TrueF = *term* (λ *s*. *True*)
apply (*simp only*: *term-def TrueF-def*)
done

constdefs
Conj::*SALform list* ⇒ *SALform*
Conj fl ≡ λ *s*. ∀ *f*∈*set fl*. *f*(*s*)

constdefs
Impl::*SALform* ⇒ *SALform* ⇒ *SALform*
Impl a b ≡ λ *s*. *a*(*s*) ⟶ *b*(*s*)

2 Weakest Preconditions

constdefs

wpF::*SALprogram* ⇒ *nat* ⇒ *nat* ⇒ *SALform* ⇒ *SALform*
wpF p i j Q ≡ *case* (*cmd p i*) *of None* ⇒ *FalseF*
| *Some ins* ⇒ (*case ins of*
SET x n ⇒ (λ(*pc*, *m*). *Q* (*j*, *m*[*x* ↦ *NAT n*]))
| *ADD x y* ⇒ (λ(*pc*, *m*). *Q* (*j*, *m*[*x* ↦ *lift* (*op* +) (*m x*) (*m y*)]))
| *INC x* ⇒ (λ(*pc*, *m*). *Q* (*j*, *m*[*x* ↦ *lift* (*op* +) (*m x*) (*NAT 1*)]))
| *JMPEQ x y t* ⇒ (λ(*pc*, *m*). *Q* (*j*, *m*))
| *JMPB t* ⇒ (λ(*pc*, *m*). *Q* (*j*, *m*)))

lemma [*code*]:
wpF p i j Q = (*case* (*cmd p i*) *of None* ⇒ *FalseF*
| *Some ins* ⇒ (*case ins of*
SET x n ⇒ *term* (λ(*pc*, *m*). (*to-term Q*) (*to-term j*, *m*[(*to-term x*) ↦ *NAT* (*to-term n*)]))
| *ADD x y* ⇒ *term* (λ(*pc*, *m*). (*to-term Q*) (*to-term j*, *m*[(*to-term x*) ↦ *lift* (*op* +) (*m* (*to-term x*)) (*m* (*to-term y*))]))
| *INC x* ⇒ *term* (λ(*pc*, *m*). (*to-term Q*) (*to-term j*, *m*[(*to-term x*) ↦ *lift* (*op* +) (*m* (*to-term x*)) (*NAT 1*)]))
| *JMPEQ x y t* ⇒ *term* (λ(*pc*, *m*). (*to-term Q*) (*to-term j*, *m*))
| *JMPB t* ⇒ *term* (λ(*pc*, *m*). (*to-term Q*) (*to-term j*, *m*))))
apply (*simp only*: *wpF-def term-def to-term-def*)
done

3 Control Flow

constdefs
succsF::*SALprogram* ⇒ *nat* ⇒ (*nat* × *SALform*) *list*
succsF p i ≡ *case* (*cmd p i*) *of*
None ⇒ []
| *Some ins* ⇒ (*case ins of*
SET x n ⇒ [(*i* + *1*, *TrueF*)]
| *ADD x y* ⇒ [(*i* + *1*, *TrueF*)]
| *INC x* ⇒ [(*i* + *1*, *TrueF*)]
| *JMPEQ x y t* ⇒ [(*i* + *t*, λ(*pc*, *m*). *m x* = *m y*), (*i* + *1*, λ(*pc*, *m*). *m x* ≠ *m y*)]
| *JMPB t* ⇒ [(*i* − *t*, *TrueF*)])

lemma [*code*]:
succsF p i = (*case* (*cmd p i*) *of*
None ⇒ []
| *Some ins* ⇒ (*case ins of*
SET x n ⇒ [(*i* + *1*, *TrueF*)]
| *ADD x y* ⇒ [(*i* + *1*, *TrueF*)]
| *INC x* ⇒ [(*i* + *1*, *TrueF*)]
| *JMPEQ x y t* ⇒ [(*i* + *t*, *term* (λ(*pc*, *m*). (*m* (*to-term x*)) = (*m* (*to-term y*)))),
(*i* + *1*, *term* (λ(*pc*, *m*). (*m* (*to-term x*)) ≠ (*m* (*to-term y*))))]
| *JMPB t* ⇒ [(*i* − *t*, *TrueF*)]))
apply (*simp only*: *succsF-def term-def to-term-def*)

done

constdefs *anF*::*SALprogram* ⇒ *nat* ⇒ (*SALform option*)
anF prg p ≡ *snd* (*prg* ! *p*)

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

constdefs
ipc :: *SALprogram* ⇒ *nat*
ipc prg ≡ *0*

4 Safety Policy

constdefs
safeF::*SALprogram* ⇒ *nat* ⇒ *SALform*
safeF prg p ≡ (*case* (*cmd prg p*) *of*
None ⇒ *FalseF*
| *Some a* ⇒ (*case a of*
SET X n ⇒ *TrueF*
| *ADD X Y* ⇒ (λ (*p*,*m*). *m X* ≠ *ILLEGAL* ∧ *m Y* ≠ *ILLEGAL*)
| *INC X* ⇒ (λ (*p*,*m*). *m X* ≠ *ILLEGAL*)
| *JMPEQ X Y t* ⇒ *TrueF*
| *JMPB t* ⇒ *TrueF*))

lemma [*code*]:
safeF prg p = (*case* (*cmd prg p*) *of*
None ⇒ *FalseF*
| *Some ins* ⇒ (*case ins of*
SET X n ⇒ *TrueF*
| *ADD X Y* ⇒ *term* (λ (*p*,*m*). *m* (*to-term X*) ≠ *ILLEGAL* ∧ *m* (*to-term Y*) ≠ *ILLEGAL*)
| *INC X* ⇒ *term* (λ (*p*,*m*). *m* (*to-term X*) ≠ *ILLEGAL*)
| *JMPEQ X Y t* ⇒ *TrueF*
| *JMPB t* ⇒ *TrueF*))
apply (*simp only*: *term-def to-term-def safeF-def*)
done

constdefs *initF*:: *SALprogram* ⇒ *SALform*
initF prg ≡ λ (*p*,*m*). *p*=*0* ∧ (∀ *X*. *m X* = *ILLEGAL*)

lemma [*code*]:
initF prg = *term* (λ (*p*,*m*). *p*=*0* ∧ (∀ *X*. *m X* = *ILLEGAL*))
apply (*simp only*: *term-def to-term-def initF-def*)
done

4.1 Wellformedness Checker

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

primrec
checkPos prg [] = *True*
checkPos prg (*p* # *ps*) = (*if* ((*case* (*cmd prg p*) *of*
None ⇒ *False*
| *Some c* ⇒ (*case c of*
SET x n ⇒ *True*
| *ADD x y* ⇒ *True*
| *INC x* ⇒ *True*
| *JMPEQ x y t* ⇒ (*t* = *0*) ⟶ ((*anF prg p*) ≠ *None*)
| *JMPB t* ⇒ ((*anF prg* (*p* − *t*)) ≠ *None*)
)))
then (*checkPos prg ps*)
else False)

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

5 Instantiating the VCG

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

ML {∗
fun prf-size () =
let
val proof-state = *Toplevel.proof-of* (*Toplevel.get-state* ());
val (-, (-, *st*)) = *Proof.get-goal proof-state*;
val {*der* = (-, *prf*), ...} = *rep-thm st*
in Proofterm.size-of-proof (*prf*)
end;
∗}

ML {∗
fun nprf-size () =
let
val proof-state = *Toplevel.proof-of* (*Toplevel.get-state* ());
val (-, (-, *st*)) = *Proof.get-goal proof-state*;
val {*der* = (-, *prf*), ...} = *rep-thm st*
in Proofterm.size-of-proof (*Proofterm.rewrite-proof-notypes* ([],

ProofRewriteRules.rprocs false) *prf*)
end;
∗}

end