theory *SALOverflowFWInst* = *SALOverflowPlatform*:

In this theory we show that the SALPlatform satisfies all assumptions of the PCC Framework.

1 Auxiliary Functions and Lemmata

lemma *isafeP-elims*:
$[\![\ s \in (isafeP\ prg);$
$\quad \bigwedge s'.\ [\![\ s = s';\ valid\ prg\ s'\ (initF\ prg)\ ]\!] \Longrightarrow P;$
$\quad \bigwedge s'\ s''.\ [\![\ s = s'';\ s' \in (isafeP\ prg);$
$\qquad valid\ prg\ s'\ (isafeF\ prg\ (fst\ s'));$
$\qquad valid\ prg\ s''\ (isafeF\ prg\ (fst\ s''));$
$\qquad (s',s'') \in (effS\ prg)$
$\quad]\!] \Longrightarrow P\ ]\!] \Longrightarrow P$**done**

lemma *isafeP-elim*:
$[\![\ (pc,m,e) \in (isafeP\ prg);\ 1 < length\ (cs\ e)\ ]\!] \Longrightarrow (valid\ prg\ (pc,m,e)\ (isafeF\ prg\ pc))$
done

lemma *isafeP-init*:
$[\![\ valid\ \ prg\ s\ (initF\ prg)\ ]\!] \Longrightarrow s \in (isafeP\ prg)$**done**

lemma *isafeP-step*:
$[\![\ \ s \in (isafeP\ prg);$
$\quad valid\ prg\ s\ (isafeF\ prg\ (fst\ s));$
$\quad valid\ prg\ s'\ (isafeF\ prg\ (fst\ s'));$
$\quad (s,s') \in (effS\ prg)\ \ ]\!] \Longrightarrow s' \in (isafeP\ prg)$**done**

lemma *noProcNoCmd*:
$[\![\ noProc\ prg\ pn]\!] \Longrightarrow cmd\ prg\ (pn,i) = None$**done**

lemma *delPos-simp*:
$(pn,i) \notin set\ (delPos\ pl\ pn)$**done**

lemma *delPos-notelem-simp*:
$pn \neq pn' \Longrightarrow ((pn,i) \in set\ (delPos\ pl\ pn')) = ((pn,i) \in set\ pl)$**done**

lemma *noProc-domC*:
$noProc\ ps\ pn \Longrightarrow (((pn,i)::pos) \notin set\ (domC\ ps))$**done**

lemma *delPos-comm*:
$(delPos\ (delPos\ pl\ a)\ b) = (delPos\ (delPos\ pl\ b)\ a)$**done**

lemma *delPos-split*:
$delPos\ (pl@pl')\ pn = (delPos\ pl\ pn)\ @\ (delPos\ pl'\ pn)$**done**

lemma *noProc-delPos*:
$\forall$ *ps pn. noProc ps pn* $\longrightarrow$ (*delPos* (*domC ps*) *pn*) = (*domC ps*)**done**

lemma *domC-simp*:
$pn \neq pn' \Longrightarrow ((pn,\ i) \in set\ (domC\ ((pn',\ bdy')\ \#\ ps))) = ((pn,\ i) \in set\ (domC\ ps))$**done**

lemma *set-list-split*:
$x \in set\ l \Longrightarrow \exists\ l1\ l2.\ l=l1@[x]@l2$**done**

lemma *domC-split*:
$cmd\ prg\ (pn,i) = Some\ instr \Longrightarrow \exists\ l1\ l2.\ (domC\ prg) = (l1@[(pn,i)]@l2)$
done

lemma *cmd-domC*:
$\forall\ instr\ pc\ prg.\ cmd\ prg\ pc = Some\ instr \longrightarrow pc \in set\ (domC\ prg)$**done**

lemma *domC-cmd*:
$\forall\ pc.\ pc \notin set\ (domC\ prg) \longrightarrow\ cmd\ prg\ pc = None$**done**

lemma *elem-set-append-cases*:
$\forall\ x\ l1\ l2.\ \ x \in set\ (l1@l2) = (x \in set\ l1) \lor (x \in set\ l2)$
by *auto*

lemma *suc-minus-swap*:
$[\![\ a <= b\ ]\!] \Longrightarrow (Suc\ b) - a = Suc\ (b - a)$
by *arith*

lemma *rec-upd-simp*:
$e(\!|\ h{:=}\ x,\ cs := y,\ h{:=}\ z|\!) = e(\!|\ cs := y,\ h{:=}\ z|\!)$
by *simp*

lemma *rec-upd-simp2*:
$e(\!|\ h{:=}\ x,\ cs := y, cs{:=}y',\ h{:=}\ z|\!) = e(\!|\ cs := y',\ h{:=}\ z|\!)$
by *simp*

lemma *rec-upd-simp3*:
$e(\!|\ cs{:=}\ x,\ cs := y|\!) = e(\!|\ cs{:=}\ y|\!)$
by *simp*

lemma *min-elim*:
$(a{::}'a{::}order) < b \Longrightarrow (min\ a\ b) = a$**done**

lemma *checkPos-split*:
$checkPos\ prg\ (l1@l2) = ((checkPos\ prg\ l1) \land (checkPos\ prg\ l2))$**done**

2 Instantiating the Safety Logic

theorem *SALSafetyLogicIns*:
*SafetyLogic Conj Impl TrueF FalseF valid***done**

lemma *isafe-imp-safeF*:
∀ *prg s. valid prg s* (*isafe* (*domC prg*, *prg*, *anF prg*, *fst s*, *FalseF*, *Conj*, *Impl*, *safeF*, *succsF*, *wpF*)) ⟶ *valid prg s* (*safeF prg* (*fst s*))**done**

lemma *isafeF-imp-safeF*:
valid prg (*pc*,*m*,*e*) (*isafeF prg pc*) ⟹ *valid prg* (*pc*,*m*,*e*) (*safeF prg pc*)**done**

3 System Invariants

Useful properties about states that originate from a safe execution.

3.1 System Invariant 1

consts *sysinv*::*SALstate* × *SALprogram* ⇒ *bool*

recdef

```
  sysinv  measure (λ ((pc,m,e),prg). length (cs e))
  sysinv ((pc,m,e),prg) = (case (cs e)
                            of [] ⇒ False
                            | c#css ⇒ (let (k,m)=c; (pn',i')=(h e)!k; (pn,i)=pc
                                        in (case css
                                             of [] ⇒ pn=0 ∧ (∀ i0 x. cmd prg (0,i0) ≠
Some (RET x))
                                             | c'#css' ⇒ (∃ x. cmd prg (pn',i') = Some
(CALL x pn))
                                                         ∧ k < length (h e)
                                                      ∧ sysinv (((pn',i'),m,e(|cs:=css,h:=take
k (h e)|)),prg)
                                           )
                                        )
                            )
```

(**hints** *recdef-simp*: *filterreduction*)

theorem *sysinv-pres*:
∀ *prg s. wf prg* ⟶ *s* ∈ (*isafeP prg*) ⟶ *sysinv*(*s*,*prg*)**done**

3.2 System Invariant 2

consts *sysinv2*::*SALstate* × *SALprogram* ⇒ *bool*

recdef

```
  sysinv2  measure (λ ((pc,m,e),prg). length (cs e))
  sysinv2 ((pc,m,e),prg) = (case (cs e)
```

of [] ⇒ *False*
| *c*#*css* ⇒ (*let* (*k*,m'')=*c*; (pn',i')=(*h e*)!*k*
in (*case css*
of [] ⇒ *True*
| c'#css' ⇒ (k < *length* (*h e*) ∧
valid prg ((pn',i'),m'',*e*(| *cs*:=*css*, *h*:=*take k* (*h e*) |)) (*isafeF prg* (pn',i')) ∧
sysinv2 (((pn',i'),m'',*e*(| *cs*:=*css*, *h*:=*take k* (*h e*) |)),*prg*)
)
)
))
(**hints** *recdef-simp*: *filterreduction*)

lemma *sysinv2-pres*:
∀ *prg s*. *wf prg* ⟶ *s* ∈ (*isafeP prg*) ⟶ *sysinv2* (*s*,*prg*)**done**

lemma *ex-weak*: $\exists\ n\ n'.\ a = NAT\ n \wedge b = NAT\ n' \wedge n = n' \Longrightarrow \exists\ n\ n'.\ a = NAT\ n \wedge b = NAT\ n'$ **by** *auto*

lemma *ex-eq*: $(\exists n.\ m\ nat1 = NAT\ n) \wedge (\exists n'.\ m\ nat2 = NAT\ n') \Longrightarrow \neg (\exists n.\ m\ nat1 = NAT\ n \wedge m\ nat2 = NAT\ n) \Longrightarrow (\exists n\ n'.\ m\ nat1 = NAT\ n \wedge m\ nat2 = NAT\ n' \wedge n \neq n')$ **by** *auto*

lemma *ex-le*: $(\exists n.\ m\ nat1 = NAT\ n) \wedge (\exists n'.\ m\ nat2 = NAT\ n') \Longrightarrow \neg (\exists n.\ m\ nat1 = NAT\ n \wedge (\exists n'.\ m\ nat2 = NAT\ n' \wedge n \leq n')) \Longrightarrow \exists n.\ m\ nat1 = NAT\ n \wedge (\exists n'.\ m\ nat2 = NAT\ n' \wedge \neg\ n \leq n')$ **by** *auto*

lemma *ex-less*: $(\exists n.\ m\ nat1 = NAT\ n) \wedge (\exists n'.\ m\ nat2 = NAT\ n') \Longrightarrow \neg (\exists n.\ m\ nat1 = NAT\ n \wedge (\exists n'.\ m\ nat2 = NAT\ n' \wedge n < n')) \Longrightarrow \exists n.\ m\ nat1 = NAT\ n \wedge (\exists n'.\ m\ nat2 = NAT\ n' \wedge \neg\ n < n')$ **by** *auto*

lemma *list-length-nth*:
(*la*@[*i*,*j*])!(*length la*)=*i***done**

lemma *list-length-nth2*:
(*la*@[*i*])!(*length la*)=*i***done**

lemma *list-length-suc-nth*:
(*la*@[*i*,*j*])!(*Suc* (*length la*))=*j***done**

lemma *path-succsF*:
⟦ *l* ∈ *paths prg succsF*; *k*+*1*<*length l* ⟧ ⟹ ∃ *B*. (*l*!(*k*+*1*),*B*) ∈ *set* (*succsF prg* (*l*!*k*))**done**

lemma *less-chain-simp*:
⟦ ∀ *k*. *length l* <= *k*+*1* ∨ *l*!*k* < *l*!(*k*+*1*); *1* < *length* (*l*::*pos list*) ⟧ ⟹ *hd l* <

*last l***done**

lemma *path-length*:
$[\![\ l \in paths\ prg\ succsF \ ]\!] \Longrightarrow 1 < length\ l$**done**

lemma *loop-has-back-jump*:
$[\![\ l \in paths\ prg\ succsF;\ hd\ l = last\ l \ ]\!] \Longrightarrow \exists\ k.\ (k+1)<length\ l \wedge l!(k+1) <= l!k$**done**

lemma *back-jumps-annotated*:
$wf\ prg \Longrightarrow \forall\ pc''\ pc\ B.\ (pc'',B) \in set\ (succsF\ prg\ pc) \longrightarrow pc''<=pc \longrightarrow (anF\ prg\ pc'') \neq None$**done**

lemma *isafeP-isafeF-initF*:
$s \in (isafeP\ prg) \Longrightarrow valid\ prg\ s\ (initF\ prg) \vee valid\ prg\ s\ (isafeF\ prg\ (fst\ s))$
apply (*erule isafeP-elims*)
apply *simp+*
done

Here we prove the framework's requirement for succsF

theorem *succsF-complete*:
$\forall\ prg\ s'\ s.\ wf\ prg \longrightarrow s \in (isafeP\ prg) \longrightarrow (s,s') \in (effS\ prg) \longrightarrow (\exists\ B.\ (fst\ s',B) \in set\ (succsF\ prg\ (fst\ s)) \wedge valid\ prg\ s\ B)$**done**

— Instantiating the VCG Framework

theorem *SAL-VCG-Ins*:
*VerificationConditionGenerator Conj Impl TrueF FalseF valid provable effS wpF succsF initF ipc safeF anF domC wf***done**

4 Platform Soundness

constdefs $isSafe::SALprogram \Rightarrow bool$
$isSafe\ prg \equiv (\forall\ s\ s'.\ prg,s \models initF\ prg \wedge (s,s') \in (effS\ prg)^* \longrightarrow prg,s' \models safeF\ prg\ (fst\ s'))$

theorem *platform-soundness*:
$[\![\ wf\ prg;\ provable\ prg\ (vcgSAL\ prg) \ ]\!] \Longrightarrow isSafe\ prg$**done**

end