theory *SALTypeSafetyFWInst = SALTypeSafetyPlatform + VerificationConditionGenerator*:

1 Proving the Safety Logic requirements

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

2 Proving the VCG requirements

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*:
$[\![\ l \in paths\ prg\ succsF;\ k+1<length\ l\ ]\!] \Longrightarrow \exists\ B.\ (l!(k+1),B) \in set\ (succsF\ prg\ (l!k))$**done**

lemma *less-chain-simp*:
$[\![\ \forall\ k.\ length\ l <= k+1 \vee l!k < l!(k+1);\ 1 < length\ (l::nat\ \ list)\ ]\!] \Longrightarrow hd\ l < last\ l$**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 *domC-split*:
$\bigwedge p.\ cmd\ prg\ p = Some\ a \Longrightarrow \exists\ l1\ l2.\ (domC\ prg) = (l1@[p]@l2)$
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 *checkPos-split*:
$checkPos\ prg\ (l1@l2) = ((checkPos\ prg\ l1) \wedge (checkPos\ prg\ l2))$**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*