theory *VCOpt = SALOverflowFWInst-deep*:

We define an optimizer for verfication condtions and prove that the transformations it does preserve validity.

consts *isconstant* ::*expr* ⇒ *bool*

primrec
isconstant (*V n*) = *False*
isconstant (*Lv v*) = *False*
isconstant (*C tv*) = *True*
isconstant PC = *False*
isconstant LastRA = *False*
isconstant Time = *False*
isconstant (*Add ex1 ex2*) = (*isconstant ex1* ∧ *isconstant ex2*)
isconstant (*Minus ex1 ex2*) = (*isconstant ex1* ∧ *isconstant ex2*)
isconstant (*Mult ex1 ex2*) = (*isconstant ex1* ∧ *isconstant ex2*)
isconstant (*Deref ex*) = *False*
isconstant (*Ifeq ex1 ex2 ex3 ex4*) = (*isconstant ex1* ∧ *isconstant ex2* ∧ *isconstant ex3* ∧ *isconstant ex4*)
isconstant (*Old ex*) = *isconstant ex*

consts *isconstantF* ::*SALform* ⇒ *bool*

recdef *isconstantF measure sizeF*
isconstantF T = *True*
isconstantF F = *True*
isconstantF (*And* []) = *True*
isconstantF (*And* (*f*#*fs*)) = (*isconstantF f* ∧ *isconstantF* (*And fs*))
isconstantF (*Imp f1 f2*) = (*isconstantF f1* ∧ *isconstantF f2*)
isconstantF (*Neg f*) = (*isconstantF f*)
isconstantF (*Eq ex1 ex2*) = (*isconstant ex1* ∧ *isconstant ex2*)
isconstantF (*Leq ex1 ex2*) = (*isconstant ex1* ∧ *isconstant ex2*)
isconstantF (*Less ex1 ex2*) = (*isconstant ex1* ∧ *isconstant ex2*)
isconstantF (*Ty ex vt*) = (*isconstant ex*)
isconstantF (*Forall n f*) = *False*

consts *flattenAnd*::*SALform* ⇒ *SALform list*
consts *flattenAndL*::*SALform list* ⇒ *SALform list*

primrec
flattenAndL [] = []
flattenAndL (*f*#*fs*) = (*flattenAnd f*)@(*flattenAndL fs*)
flattenAnd T = []
flattenAnd F = [*F*]
flattenAnd (*And fs*) = (*flattenAndL fs*)
flattenAnd (*Imp f1 f2*) = [*Imp f1 f2*]
flattenAnd (*Neg f*) = [*Neg f*]
flattenAnd (*Eq ex1 ex2*) = [*Eq ex1 ex2*]
flattenAnd (*Leq ex1 ex2*) = [*Leq ex1 ex2*]

$flattenAnd\ (Less\ ex1\ ex2) = [Less\ ex1\ ex2]$
$flattenAnd\ (Ty\ ex\ vt) = [Ty\ ex\ vt]$
$flattenAnd\ (Forall\ n\ f) = [Forall\ n\ f]$

constdefs $I0::nat \Rightarrow tval$
$I0 \equiv \lambda\ x.\ ILLEGAL$

consts $vcopt::SALform\ list \Rightarrow SALform \Rightarrow SALform$
$vcoptL::SALform\ list \Rightarrow\ SALform\ list \Rightarrow SALform\ list$

primrec
$vcoptL\ G\ [] = []$

$vcoptL\ G\ (f\#fs) = (vcopt\ G\ f)\#(vcoptL\ G\ fs)$

$vcopt\ G\ T = T$

$vcopt\ G\ F = F$

$vcopt\ G\ (And\ fs) = (case\ (And\ fs\ mem\ G)$
$\quad of\ True \Rightarrow T$
$\quad | \ False \Rightarrow (let\ fs2 = (delL\ T\ (flattenAndL\ (vcoptL\ G\ fs)))$
$\quad\quad in\ (case\ fs2$
$\quad\quad\quad of\ [] \Rightarrow T$
$\quad\quad\quad | \ (f2'\#fs2') \Rightarrow (case\ (F\ mem\ fs2)$
$\quad\quad\quad\quad\quad of\ True \Rightarrow F$
$\quad\quad\quad\quad\quad | \ False \Rightarrow And\ fs2))))$

$vcopt\ G\ (Imp\ f1\ f2) = (case\ (Imp\ f1\ f2\ mem\ G)$
$\quad of\ True \Rightarrow T$
$\quad | \ False \Rightarrow (let\ f1' = vcopt\ G\ f1;$
$\quad\quad\quad G' = G\ @\ (flattenAnd\ f1');$
$\quad\quad\quad f2' = vcopt\ G'\ f2$
$\quad\quad in\ (case\ (f1' = T)$
$\quad\quad\quad of\ True \Rightarrow f2'$
$\quad\quad\quad | \ False \Rightarrow$
$\quad\quad\quad (case\ (f1' = F)$
$\quad\quad\quad of\ True \Rightarrow T$
$\quad\quad\quad | \ False \Rightarrow$
$\quad\quad\quad (case\ (f2' = T)$
$\quad\quad\quad of\ True \Rightarrow T$
$\quad\quad\quad | \ False \Rightarrow Imp\ f1'\ f2')))))$

$vcopt\ G\ (Neg\ f) = (let\ f' = vcopt\ G\ f$
$\quad\quad in\ (case\ (f' = T)$
$\quad\quad of\ True \Rightarrow F$
$\quad\quad | \ False \Rightarrow (case\ (f' = F)$
$\quad\quad\quad\quad of\ True \Rightarrow T$
$\quad\quad\quad\quad | \ False \Rightarrow Neg\ f')))$

```
vcopt G (Eq ex1 ex2) = (case (Eq ex1 ex2 mem G)
                        of True ⇒ T
                        | False ⇒
                         (case (isconstant ex1 & isconstant ex2)
                        of True ⇒ (case (eval I0 initialState ex1 = (eval I0 initialState
ex2))
                                        of True ⇒ T
                                        | False ⇒ F)
                          |  False ⇒ Eq ex1 ex2))

vcopt G (Leq ex1 ex2) = (case (Leq ex1 ex2 mem G)
                        of True ⇒ T
                        | False ⇒
                         (case (isconstant ex1 & isconstant ex2)
                          of True ⇒ (case (nv (eval I0 initialState ex1) ≤ nv (eval I0
initialState ex2))
                                        of True ⇒ T
                                        | False ⇒ F)
                         | False ⇒ Leq ex1 ex2))

vcopt G (Less ex1 ex2) = (case (Less ex1 ex2 mem G)
                        of True ⇒ T
                        | False ⇒
                         (case (isconstant ex1 & isconstant ex2)
                          of True ⇒ (case (nv (eval I0 initialState ex1) < nv (eval I0
initialState ex2))
                                        of True ⇒ T
                                        | False ⇒ F)
                         | False ⇒ Less ex1 ex2))


vcopt G (Ty ex vt) = (case (Ty ex vt mem G)
                      of True ⇒ T
                      | False ⇒ (case (isconstant ex)
                                  of True ⇒ (case (ty (eval I0 initialState ex) = vt)
                                              of True ⇒ T
                                              | False ⇒ F)
                                  | False ⇒ Ty ex vt))
vcopt G (Forall v f) = (case (Forall v f mem G)
                        of True ⇒ T
                        | False ⇒ Forall v f)
```

0.1 Soundness of the optimizer

lemma *delT-And-validF*:
(*validF I s* (*And* (*delL T fs*))) = (*validF I s* (*And fs*))**done**

lemma *delT-validFs*:

$(validFs\ I\ s\ (delL\ T\ fs)) = (validFs\ I\ s\ fs)$**done**

lemma *validF-And-split*:
$(validF\ I\ s\ (And\ (fs1\ @\ fs2))) = (validF\ I\ s\ (And\ fs1) \wedge validF\ I\ s\ (And\ fs2))$**done**

lemma *validFs-split*:
$(validFs\ I\ s\ (fs1\ @\ fs2)) = (validFs\ I\ s\ fs1 \wedge validFs\ I\ s\ fs2)$**done**

lemma *flattenAnd-validF*:
$(validF\ I\ s\ (And\ (flattenAnd\ f))) = (validF\ I\ s\ f)$
done

lemma *flattenAnd-validFs*:
$(validFs\ I\ s\ (flattenAnd\ f)) = (validF\ I\ s\ f)$
done

lemma *flattenAndL-validF*:
$validF\ I\ s\ (And\ (flattenAndL\ fs)) = validF\ I\ s\ (And\ fs)$**done**

lemma *flattenAndL-validFs*:
$validFs\ I\ s\ (flattenAndL\ fs) = validFs\ I\ s\ fs$**done**

lemma *delL-split*:
$delL\ a\ (l1\ @\ l2) = (delL\ a\ l1)\ @\ (delL\ a\ l2)$**done**

lemma *F-And*:
$F \in set\ fs \Longrightarrow \neg\ (validF\ I\ s\ (And\ fs))$**done**

lemma *F-invalid*:
$F \in set\ fs \Longrightarrow \neg\ (validFs\ I\ s\ fs)$**done**

lemma *vcoptL-src*:
$\bigwedge x\ y.\ (x \in (set\ (vcoptL\ G\ fs))) \Longrightarrow (\exists\ y \in (set\ fs).\ x = (vcopt\ G\ y))$**done**

lemma *not-And-validF*:
$\neg\ (validF\ I\ s\ (And\ fs)) \Longrightarrow (\exists\ f \in set\ fs.\ \neg\ (validF\ I\ s\ f))$**done**

lemma *And-validF-all*:
$validF\ I\ s\ (And\ fs) = (\forall\ f \in set\ fs.\ validF\ I\ s\ f)$**done**

lemma *isconstant-eval*:
$\bigwedge s\ s'.\ isconstant\ ex \Longrightarrow eval\ I\ s\ ex = eval\ I'\ s'\ ex$**done**

lemma *isconstant-validF*:
$\bigwedge f.\ isconstantF\ f \Longrightarrow (validF\ I\ s\ f) = (validF\ I'\ s'\ f)$**done**

declare *split-paired-All* [*simp del*] *split-paired-Ex* [*simp del*]
ML-setup {∗

simpset-ref() := *simpset*() *delloop split-all-tac*;
claset-ref () := *claset* () *delSWrapper split-all-tac*
∗}

lemma *vcopt-sound*:
$\bigwedge f\ G\ (I{::}var \Rightarrow tval).\ (\forall\ f' \in set\ G.\ I,s \models f') \Longrightarrow (I,s \models vcopt\ G\ f) = (I,s \models f)$**done**

theorem *vcopt-soundness*:
$((prg{::}SALprogram),s \models vcopt\ []\ f) = (prg,s \models f)$**done**

end