theory *DeepLogic* = *FiniteMap* + *SALSyntax*:

1 Syntax and Manipulation of formulae

datatype
tval = *ILLEGAL* | *NAT nat* | *POS pos*

constdefs
MAX :: *nat*
MAX ≡ *15*

datatype
vtype= *Nat* | *Pos*

types *var* = *nat*

datatype *expr* = *V var* | *Lv var* |
C tval |
Pc |
Rp |
Tm |
Add expr expr (**infixr** ⊕ *65*) |
Minus expr expr (**infixr** ⊖ *65*) |
Mult expr expr (**infixr** ⊙ *65*) |

Deref expr |
Ifeq expr expr expr expr (*IF* - ≐ - *THEN* - *ELSE* - [*61*,*61*,*60*,*60*] *60*) |
Old expr

datatype *form* = *T* |
F |
And form list (Λ - *70*) |
Imp form form (**infix** ⊃ *65*) |
Neg form |
Eq expr expr (**infixr** ≐ *65*) |
Leq expr expr (**infixr** ≼ *65*) |
Less expr expr (**infixr** ≺ *65*) |
Ty expr vtype |
Forall var form ((*3*∏ -./ -) [*0*, *10*] *10*) |
Yields expr tval list

datatype *mode* = *NoPt* | *Mv var var*

consts
substE::*mode* ⇒ (*expr* ~~> *expr*) ⇒ *expr* ⇒ *expr*

consts

```
changedvars :: (expr ~~> expr) ⇒ (var ~~> expr)

primrec
changedvars [] = []
changedvars (m#ms) = (case (fst m)
                      of V v ⇒ [(v,snd m)]
                      | Lv v ⇒ []
                      | C tv ⇒ []
                      | Pc ⇒ []
                      | Rp ⇒ []
                      | Tm ⇒ []
                      | Add e1 e2 ⇒ []
                      | Minus e1 e2 ⇒ []
                      | Mult e1 e2 ⇒ []
                      | Deref e ⇒ []
                      | Ifeq e1 e2 e3 e4 ⇒ []
                      | Old e ⇒ [])@(changedvars ms)



primrec substE:
substE m em (V v) = (case m
                      of NoPt ⇒ em ?= (V v)
                      | Mv s t ⇒ (Ifeq (V t) (C (NAT v)) (Deref (V s)) (em ?= (V
v))))
substE m em (Lv v) = (Lv v)
substE m em (C tv) = (C tv)
substE m em Pc = em ?= Pc
substE m em Rp = em ?= Rp
substE m em Tm = em ?= Tm
substE m em (Add e1 e2) = Add (substE m em e1) (substE m em e2)
substE m em (Minus e1 e2) = Minus (substE m em e1) (substE m em e2)
substE m em (Mult e1 e2) = Mult (substE m em e1) (substE m em e2)
substE m em (Deref e1) =  (let e1'= (substE m em e1);
                           res=(foldl (λ e (a,b). (Ifeq e1' (C (NAT a)) b e)) (Deref
e1') (changedvars em))                                    in (case m
                              of NoPt ⇒ res
                              | Mv s t ⇒ Ifeq (V t) e1' (Deref (V s)) res))
substE m em (Ifeq e1 e2 e3 e4) = Ifeq (substE m em e1) (substE m em e2) (substE
m em e3) (substE m em e4)
substE m em (Old e1) = Old e1

consts
intVarsE :: expr ⇒ var list

primrec
intVarsE (V n) = []
intVarsE (Lv v) = [v]
```

intVarsE (*C tv*) = []
intVarsE Pc = []
intVarsE Rp = []
intVarsE Tm = []
intVarsE (*Add e1 e2*) = (*intVarsE e1*) @ (*intVarsE e2*)
intVarsE (*Minus e1 e2*) = (*intVarsE e1*) @ (*intVarsE e2*)
intVarsE (*Mult e1 e2*) = (*intVarsE e1*) @ (*intVarsE e2*)
intVarsE (*Deref e1*) = *intVarsE e1*
intVarsE (*Ifeq e1 e2 e3 e4*) = (*intVarsE e1*) @ (*intVarsE e2*) @ (*intVarsE e3*) @ (*intVarsE e4*)
intVarsE (*Old e*) = *intVarsE e*

consts
freeIntVars :: *form* ⇒ *var list*
freeIntVarsL :: *form list* ⇒ *var list*

primrec
freeIntVarsL [] = []
freeIntVarsL (*f*#*fs*) = (*freeIntVars f*)@(*freeIntVarsL fs*)
freeIntVars T = []
freeIntVars F = []
freeIntVars (*And fs*) = *freeIntVarsL fs*
freeIntVars (*Imp f1 f2*) = (*freeIntVars f1*) @ (*freeIntVars f2*)
freeIntVars (*Neg f*) = *freeIntVars f*
freeIntVars (*Eq e1 e2*) = (*intVarsE e1*) @ (*intVarsE e2*)
freeIntVars (*Leq e1 e2*) = (*intVarsE e1*) @ (*intVarsE e2*)
freeIntVars (*Less e1 e2*) = (*intVarsE e1*) @ (*intVarsE e2*)
freeIntVars (*Ty e tv*) = (*intVarsE e*)
freeIntVars (*Forall n f*) = *filter* (λ *v*. *v*≠ *n*) (*freeIntVars f*)
freeIntVars (*Yields e vs*) = (*intVarsE e*)

constdefs *del-id-rns*::(*var* ~~> *expr*) ⇒ (*var* ~~> *expr*)
del-id-rns rm == *filter* (λ (*v*,*e*). *e* ≠ *V v*) *rm*

consts
modevars:: *mode* ⇒ *var list*
primrec
modevars NoPt = []
modevars (*Mv s t*) = [*s*,*t*]

consts
substF::*mode* ⇒ (*expr* ~~> *expr*) ⇒ *form* ⇒ *form*
substFL::*mode* ⇒ (*expr* ~~> *expr*) ⇒ *form list* ⇒ *form list*

primrec
substFL m em [] = []
substFL m em (*f*#*fs*) = (*substF m em f*)#(*substFL m em fs*)

$substF\ m\ em\ T = T$
$substF\ m\ em\ F = F$
$substF\ m\ em\ (And\ fs) = And\ (substFL\ m\ em\ fs)$
$substF\ m\ em\ (Imp\ f1\ f2) = Imp\ (substF\ m\ em\ f1)\ (substF\ m\ em\ f2)$
$substF\ m\ em\ (Neg\ f) = Neg\ (substF\ m\ em\ f)$
$substF\ m\ em\ (Eq\ e1\ e2) = (Eq\ (substE\ m\ em\ e1)\ (substE\ m\ em\ e2))$
$substF\ m\ em\ (Leq\ e1\ e2) = (Leq\ (substE\ m\ em\ e1)\ (substE\ m\ em\ e2))$
$substF\ m\ em\ (Less\ e1\ e2) = (Less\ (substE\ m\ em\ e1)\ (substE\ m\ em\ e2))$
$substF\ m\ em\ (Ty\ ex\ vt) = (Ty\ (substE\ m\ em\ ex)\ vt)$
$substF\ m\ em\ (Forall\ v\ f) = (Forall\ v\ (substF\ m\ em\ f))$
$substF\ m\ em\ (Yields\ ex\ vs) = (Yields\ (substE\ m\ em\ ex)\ vs)$

constdefs
$nv{::}tval \Rightarrow nat$
$nv\ v \equiv (case\ v\ of\ ILLEGAL \Rightarrow (MAX{+}1) \mid NAT\ n \Rightarrow n \mid POS\ r \Rightarrow (MAX{+}1))$

constdefs
$rv{::}tval \Rightarrow pos$
$rv\ v \equiv (case\ v\ of\ ILLEGAL \Rightarrow (0,0) \mid NAT\ n \Rightarrow (0,0) \mid POS\ r \Rightarrow r)$

end