Università degli Studi di Padova

Scuola Galileiana di Studi Superiori

Classe di Scienze Naturali

Tesi di Diploma Galileiano

Towards an implementation in LambdaProlog of the two level Minimalist Foundation

Relatore
Prof. Claudio Sacerdoti Coen

Diplomando
Alberto Fiori

Contents

A General Predicates

```
%%— dependents products: setPi
type setPi mttType →
     (mttTerm → mttType) →
     mttType.
type lambda mttType →
     (mttTerm → mttTerm) →
     mttTerm.
type app mttTerm →
     mttTerm →
     mttTerm.

%%— propositional conjunction: and
type and mttType →
     mttType →
     mttType.
type pair_and mttType →
     mttType →
     mttTerm →
     mttTerm →
     mttTerm.
type p1_and, p2_and mttTerm → mttTerm.


%%— dependet sums: setSigma

type setSigma mttType →
     (mttTerm → mttType) →
     mttType.
type pair mttType →
     (mttTerm → mttType) →
     mttTerm → mttTerm →
     mttTerm.
type elim_setSigma mttTerm →
     (mttTerm → mttType) →
     (mttTerm → mttTerm → mttTerm) →
     mttTerm.

%%— existential quantifier: exist
type exist mttType →
     (mttTerm → mttType) →
     mttType.
type pair_exist mttType →
     (mttTerm → mttType) →
     mttTerm →
     mttTerm →
     mttTerm.
```

```

type elim_exist mttTerm →
     mttType →
     (mttTerm → mttTerm → mttTerm) →
     mttTerm.

%%— universal quantifier: forall
type forall mttType →
     (mttTerm → mttType) →
     mttType.
type forall_lam mttType →
     (mttTerm → mttTerm) →
     mttTerm.
type forall_app mttTerm →
     mttTerm →
     mttTerm.

%%— intensional propositional equality: propId
type propId mttType →
     mttTerm →
     mttTerm →
     mttType.
type id mttType →
     mttTerm →
     mttTerm.
type elim_id mttTerm →
     (mttTerm → mttTerm → mttType) →
     (mttTerm → mttTerm) →
     mttTerm.

%%— implication: implies
type implies mttType →
     mttType →
     mttType.
type impl_lam mttType →
     (mttTerm → mttTerm) →
     mttTerm.
type impl_app mttTerm →
     mttTerm →
     mttTerm.

%%— local definitions: letIn
type letIn mttType
     → mttTerm
     → (mttTerm → mttTerm)
     → mttTerm
     .

%%— propositional disjunction: or
```

```
type or mttType → mttType → mttType.
type inl_or, inr_or mttType
     → mttType
     → mttTerm
     → mttTerm
     .
type elim_or mttType
     → mttTerm
     → (mttTerm → mttTerm)
     → (mttTerm → mttTerm)
     → mttTerm
     .

%%— disjoint sum: setSum
type setSum mttType → mttType → mttType.
type inl, inr mttType →
     mttType →
     mttTerm →
     mttTerm.
type elim_setSum (mttTerm → mttType) →
     mttTerm →
     (mttTerm → mttTerm) →
     (mttTerm → mttTerm) →
     mttTerm.

%%— singleton set/unit type: singleton
type  singleton mttType.
type  star mttTerm.
type  elim_singleton mttTerm →
     (mttTerm → mttType) →
     mttTerm → mttTerm.

/* UNITYPED COMPUTATIONAL PREDICATES */
type hstep, dconv, hnf, conv, interp A → A → prop.

/*MTT PREDICATES*/
kind mttTerm, mttType, mttKind, mttLevel type.
type ext, int mttLevel.
type col, set, propc, props mttKind.


type locDecl mttTerm → mttType → prop.
type locDeclType mttType → mttKind → prop.
type ofType mttType → mttKind → mttLevel → prop.
type of, isa mttTerm → mttType → mttLevel → prop.

type locDef mttTerm → mttType → mttTerm → prop.

type forall mttType →
```

```
    (mttTerm → mttType) →
    mttType.

hnf A B ⊢ hstep A C, !, hnf C B.
hnf A A.

conv A A ⊢ ! .
conv A B ⊢ (locDecl _ (propEq _ A B) ).
conv A B
    ⊢ spy(hnf A A')
    , spy( hnf B B')
    , spy( dconv A' B')
    .

dconv A A ⊢ !.

pts_leq A A.
pts_leq props set.
pts_leq props col.
pts_leq props propc.
pts_leq set col.
pts_leq propc col.


pts_prop props props props ⊢ !.
pts_prop _ _ propc.

pts_fun A B set
    ⊢ spy(pts_leq A set)
    , spy(pts_leq B set)
    , !
    .
pts_fun _ _ col.

pts_for A props props ⊢ pts_leq A set, !.
pts_for _ _ propc.

%ofType A KIND IE ⊢ locDeclType A KIND.

isaType Type Kind IE
    ⊢ spy(ofType Type Kind' IE)
    , spy(pts_leq Kind' Kind)
    .

of (fixMe2 M T ) T int
    ⊢ !
    , print "|<| Found a FixMe! |>|"
    , print M
    , term_to_string T S, print S
```

```
    .

isa (fixMe M) T int
    ⊢ !
    , print "|<| Found a FixMe! |>|"
    , print M
    , term_to_string T S, print S
    .


isa Term TY IE
    ⊢ spy(of Term TY' IE)
    , spy(conv TY' TY)
    .


of X Y _ ⊢ locDecl X Y .


tau_proof_eq A A T H'
    ⊢ interp_isa A T Ai
    , setoid_refl T H
    , H' = H Ai
    .

tau_proof_eq A B T Hi
    ⊢ spy(locDecl H (propEq T' A B)), !
    , spy(interp_isa H (propEq T A B) Hi)
    .

tau_proof_eq A B T Hi
    ⊢ (locDecl H (propEq T' B A))
    , spy(interp_isa H (propEq T B A) Hi')
    , spy (setoid_symm T Q)
    , Hi = Q Hi'
    .



tau A A (x \ x) ⊢ !.

tau_trasp A A (x\y\h\ h) ⊢ !.


%interpret X:_ext T in un Xi di tipi Ti
interp_isa X T Xi
    ⊢ spy(of X T_inf ext)
    , spy(interp X Xi')
    , spy(tau T_inf T F)
```

```
    , spy(Xi = F Xi')
    .

locDecl (k_propId Te)
    (forall T t1\
        forall T t1'\
        implies (E t1 t1')
            (forall T t2\ forall T t2'\
                implies (E t2 t2')
                    (implies (E t1 t2)
                        (E t1' t2'))))
    ⊢ interp Te T
    , setoid_eq Te E
    .

setoid_symm T (x\ fixMe "prova di symmetria").

macro_tau B B' Q
    ⊢ spy(setoid_eq B EquB)
    , spy(interp B Bi)
    , spy(interp B' Bi')
    , spy(pi x1 \ pi x2 \ pi h\
       pi x1i\ pi x2i\ pi hi\
          locDecl x1 B
        ⇒ locDecl x2 B'
        ⇒ locDecl x1i Bi
        ⇒ locDecl x2i Bi'
        ⇒ interp x1 x1i
        ⇒ interp x2 x2i
        ⇒ (locDecl h (propEq B x1 x2))
        ⇒ (locDecl hi (EquB x1i x2i))
        ⇒ interp h hi
        ⇒ spy(Q x1 x2 h x1i x2i hi)
       )
    .
macro_interp B Q ⊢ macro_tau B B Q.
```

elpi/main.elpi

B Dependent Products

```
ofType (setPi B C) KIND3 IE
    ⊢ (ofType B KIND1 IE)
    , (pi x\ locDecl x B
       ⇒ (ofType (C x) KIND2 IE))
    , spy(pts_fun KIND1 KIND2 KIND3)
    .

of (lambda B F) (setPi B C) IE
    ⊢ spy (ofType B _ IE)
    , spy (pi x\ locDecl x B ⇒ isa (F x) (C x) IE)
    .

of (app Lam X) (CX) IE
    ⊢ spy(of Lam (setPi B C) IE)
    , spy(isa X B IE)
    , CX = C X
    .


hstep (app LAM Bb) (F Bb)
    ⊢ of LAM (setPi B C) IE
    , (ofType B _ IE)
    , (isa Bb B IE)
    , hnf LAM (lambda B' F)
    , conv B B'
    , (pi x\ locDecl x B ⇒ isa (F x) (C x) IE)
    , (pi x\ locDecl x B ⇒ ofType (C x) _ IE)
    .

dconv (setPi B C) (setPi B' C')
    ⊢ (conv B B')
    , (pi x\ locDecl x B ⇒ conv (C x) (C' x))
    .
dconv (app F X) (app F' X')
    ⊢ (conv F F')
    , (conv X X')
    .

dconv (lambda B F) (lambda B' F')
    ⊢ (conv B B')
    , pi x\  locDecl x B ⇒ (conv (F x) (F' x))
    .

interp (setPi B C) T
```

```
    ⊢ spy(interp B Bi)
    , spy(pi x\ pi xi\ locDecl x B
       ⇒ locDecl xi Bi
       ⇒ interp x xi
       ⇒ interp (C x) (Ci xi))
    , spy(setoid_eq B EquB)
    , spy(pi x\ pi xi\ locDecl x B
       ⇒ locDecl xi Bi
       ⇒ interp x xi
       ⇒ setoid_eq (C x) (EquC xi))
    , spy(pi x1 \ pi x2 \ pi h\
        pi x1i\ pi x2i\ pi hi\ locDecl x1 B
       ⇒ locDecl x2 B
       ⇒ locDecl x1i Bi
       ⇒ locDecl x2i Bi
       ⇒ interp x1 x1i
       ⇒ interp x2 x2i
       ⇒ (locDecl h (propEq B x1 x2))
       ⇒ (locDecl hi (EquB x1i x2i))
       ⇒ interp h hi
       ⇒ spy(tau (C x1) (C x2)
            (TauC x1i x2i hi)))
    , T = setSigma (setPi Bi Ci) f\
        (forall (Bi) x1\
        (forall Bi x2\
        (forall (EquB x1 x2) h\
        (EquC x2
            (TauC x1 x2 h (app f x1))
            (app f x2)))))
    .


interp (app F X) R
    ⊢ spy(of F (setPi B C) ext)
    , spy(interp_isa X B Xi)
    , spy(interp F Fi)
    , spy(of Fi T int)
    , spy(T = (setSigma PI _))
    , R = (app
        (elim_setSigma Fi (_\PI) (x\y\x) )
        Xi)
    .


interp (lambda B F) R
    ⊢ spy(of (lambda B F) (setPi B C) ext)
    , spy(interp (setPi B C)
        (setSigma (setPi Bi Ci) H ))
    , macro_interp B
```

```
        ( x\_\_\xi\_\_\ interp (F x) (Fi xi))
    , setoid_eq B EquB
    , macro_interp B
        (x1\x2\h\x1i\x2i\hi\
            tau_proof_eq (F x1) (F x2) (C x2)
                (K_EQU x1i x2i hi))
    , R = pair (setPi Bi Ci) (H) (lambda Bi Fi)
        (forall_lam Bi x1\
        forall_lam Bi x2\
        forall_lam (EquB x1 x2) h\
            K_EQU x1 x2 h)
    .


setoid_eq (setPi B C) P
    ⊢ spy(interp B Bi)
    , spy(pi x\ pi xi\ locDecl x B
        ⇒ locDecl xi Bi
        ⇒ interp x xi
        ⇒ interp (C x) (Ci xi))
    , spy(pi x\ pi xi\locDecl x B
        ⇒ locDecl xi Bi
        ⇒ interp x xi
        ⇒ (interp (C x) (Ci xi)
            , setoid_eq (C x) (EquC xi)))
    , P = (f\ g\
        forall Bi x\
        EquC x
        (app (elim_setSigma f
            (_\setPi Bi Ci) (x\y\x) ) x)
        (app (elim_setSigma g
            (_\setPi Bi Ci) (x\y\x) ) x))
    .

tau_proof_eq (app F X1) (app F X2) T H
    ⊢ of F (setPi B T') ext
    , spy(tau_proof_eq X1 X2 B G)
    , spy(interp F Fi)
    , spy(of Fi (setSigma TyF MorF) int)
    , PI1 = (c\ elim_setSigma c
        (_\ TyF) (x \ y\ x))
    , P2Fi = elim_setSigma Fi
        (c \ MorF (PI1 c))
        (x\ y\ y)
    , spy(interp_isa X1 B X1i)
    , spy(interp_isa X2 B X2i)
    , spy(tau
        (propEq (T' X2) (app F X1) (app F X2))
        (propEq (T) (app F X1) (app F X2))
```

```
        TAU)
    ,  H = TAU
         (forall_app
         (forall_app
         (forall_app P2Fi X1i) X2i) G)
    .

tau (setPi B C) (setPi B' C') P
    ⊢ spy(interp (setPi B C) (setSigma T1 T2))
    ,  spy(T1 = setPi Bi Ci)
    ,  spy(interp (setPi B' C') (setSigma T1' T2'))
    ,  spy(T1' = setPi Bi' Ci')
    ,  spy(setoid_eq B' EquB')
    ,  spy(macro_interp B
         (_\x2\_\_\x2i\_\
             setoid_eq (C x2) (EquC x2i)))
    ,  spy(tau B' B FB)
    ,  spy(macro_tau B B'
         (x\x'\_\xi\xi'\_\
             tau (C x) (C' x') (FC' xi xi')))
    ,  spy(macro_interp B (x1\x2\_\x1i\x2i\_\
         tau (C x1) (C x2) (FCC x1i x2i)))
    ,  spy(tau_trasp B' B KB)
    ,  spy(macro_tau B B' x\x'\_\xi\xi'\hi\
         tau_trasp (C x) (C' x') (KC' xi xi' hi))
    ,  P = (w\ elim_setSigma w
         (_\setSigma T1' T2') f\p\
         pair T1' T2'
         (lambda Bi' x\ FC' (FB x) x (app f (FB x)))
         (forall_lam Bi' y1'\
         forall_lam Bi' y2'\
         forall_lam (EquB' y1' y2') d'\
        KC' (FB y2')
             y2'
             d'
             (FCC (FB y1') (FB y2') (app f (FB y1')))
             (app f (FB y2'))
             (forall_app
                 (forall_app
                 (forall_app p (FB y1'))
                 (FB y2'))
                 (KB y1' y2' d'))))
    .

tau_trasp (setPi B C) (setPi B' C') P
    ⊢ spy(macro_tau B B' x\x'\_\xi\xi'\hi\
         tau_trasp (C x) (C' x') (KC' xi xi' hi))
    ,  spy(tau B' B FB)
    ,  P = f\g\d\ forall_lam B' y'\
```

```
          KC' (FB y')
              y'
              d
              (app f (FB y'))
              (app g (FB y'))
              (forall_app d (FB y'))

    .
```

elpi/calc/setPi.elpi

C Propositional Equalities

Extensional

```
%% calc_Eq.elpi


type propEq mttType → mttTerm →
     mttTerm → mttType.
type eq    mttType →
     mttTerm → mttTerm.

pts_eq K props ⊢ pts_leq K set, !.
pts_eq _ propc.

ofType (propEq A AA1 AA2) KIND ext
     ⊢ ofType A KIND' ext
     ,  pts_eq KIND' KIND
     ,  isa AA1 A ext
     ,  isa AA2 A ext
     .


of (eq C Cc) (propEq C Cc Cc) ext
     ⊢  spy(of Cc C ext)
     .



%dstep A B ⊢ of _ ()

dconv (propEq A AA1 AA2)
      (propEq A' AA1' AA2')
     ⊢ spy(conv A A')
     ,  spy(conv AA1 AA1')
     ,  spy (conv AA2 AA2')
     .

dconv (eq A AA) (eq A' AA')
     ⊢ conv A A'
     ,  conv AA AA'
     .

interp (propEq A Aa1 Aa2) R
     ⊢ spy(setoid_eq A EquA)
     ,  spy(interp_isa Aa1 A Aa1')
     ,  spy(interp_isa Aa2 A Aa2')
     ,  spy(R = (EquA Aa1' Aa2'))
```

```
    .

interp (eq A Aa) T
    ⊢ spy(setoid_refl A ReflA)
    , spy(interp Aa Aa')
    , T = (ReflA Aa')
    .

setoid_refl (propEq _ _ _)
            (_\id singleton star).
setoid_eq (propEq A Aa1 Aa2)
          (_\ _\ (propId singleton star star)).

tau (propEq T_ T1 T2) (propEq T T1' T2') (F)
    ⊢ spy(tau_proof_eq T1 T1' T F1)
    , spy(tau_proof_eq T2 T2' T F2)
    , spy(interp_isa T1 T T1i)
    , spy(interp_isa T2 T T2i)
    , spy(interp_isa T1' T T1i')
    , spy(interp_isa T2' T T2i')
    , spy(interp T Ti)
    , F = x\ impl_app (
        impl_app (
        forall_app (
        forall_app (
        impl_app (
        forall_app (
        forall_app (k_propId T)
            T1i)
            T1i')
            F1)
            T2i)
            T2i')
            F2) x
    .
tau_trasp (propEq _ _ _ )
    (propEq _ _ _)
    (h\h'\k\ k).

tau_proof_eq _ _ (propEq T A B)
    (id singleton star).
```

elpi/calc/propEq.elpi

Intensional

```
ofType (propId A AA1 AA2) KIND IE
```

```
    ⊢ isa AA1 A int
    ,  isa AA2 A int
    ,  ofType A KIND1 int
    ,  (spy(pts_leq KIND1 set , KIND = props), !
    ;  KIND = propc)
    .

of (id A AA) (propId A AA AA) int
    ⊢ ofType A _ int
    ,  isa AA A int
    .

of (elim_id P C CC) (C AA1 AA2) int
    ⊢ (of P (propId A AA1 AA2) int)
    , (pi x\ pi y\ locDecl x A
        ⇒ locDecl y A
        ⇒ isaType (C x y) propc int)
    , (pi x\ locDecl x A
        ⇒ of (CC x) (C x x) int)
    .

hstep (elim_id (id A AA) C CC) (CC AA)
    ⊢ (isa AA A int)
    ,  (pi x\ pi y\ locDecl x A
        ⇒ locDecl y A
        ⇒ isaType (C x y) propc int)
    ,  (pi x\ locDecl x A
        ⇒ of (CC x) (C x x) int)
    .

dconv (id A AA) (id A' AA')
    ⊢ (conv A A')
    ,  (conv AA AA')
    .

dconv (propId A AA1 AA2) (propId A' AA1' AA2')
    ⊢ spy (conv A A')
    ,  spy (conv AA1 AA1')
    ,  spy (conv AA2 AA2')
    .
```

elpi/calc/propId.elpi