```
REFLECTION
----------

*C reflection_begin           ************ REFLECTION ************
*C begin_term                 ---------- term ----------
*D term                       EdAlias Term ::  Term== term
*R termFormation
               H ⊢ 𝕌 ext Term

                 BY termFormation ()

                 No Subgoals
*R termEquality
               H ⊢ Term = Term ∈ 𝕌

                 BY termEquality ()

                 No Subgoals
*M init_term   let TermEquality = (Refine ‘termEquality‘ []) ;;
               update_Trivial_additions ‘TermEquality‘ TermEquality ;;
*T term_wf                   1 Term ∈ 𝕌
*C begin_is_subst              ---------- is_subst ----------
*D is_subst0                   EdAlias is0 ::  is_subst0(.<t:term:E>)== is_subst(<t>)
*D is_subst1   EdAlias is1 ::  is_subst1(<x1:var>.<t:term:E>)== is_subst(<x1>.<t>)
*D is_subst2   EdAlias is2 ::
                 is_subst2(<x1:var>,<x2:var>.<t:term:E>)
                 == is_subst(<x1>,<x2>.<t>)
*D is_subst3   EdAlias is3 ::
                 is_subst3(<x1:var>,<x2:var>,<x3:var>.<t:term:E>)
                 == is_subst(<x1>,<x2>,<x3>.<t>)
*D is_subst4   EdAlias is4 ::
                 is_subst4(<x1:var>,<x2:var>,<x3:var>,<x4:var>.<t:term:E>)
                 == is_subst(<x1>,<x2>,<x3>,<x4>.<t>)
*D is_subst5   EdAlias is5 ::
                 is_subst5(<x1:var>,<x2:var>,<x3:var>,<x4:var>,<x5:var>.<t:term:E>)
                 == is_subst(<x1>,<x2>,<x3>,<x4>,<x5>.<t>)
*D is_subst6   EdAlias is6 ::
                 is_subst6(<x1:var>,<x2:var>,<x3:var>,<x4:var>,<x5:var>,<x6:var>.<t:term:E>)
                 == is_subst(<x1>,<x2>,<x3>,<x4>,<x5>,<x6>.<t>)
*D is_subst7   EdAlias is7 ::
                 is_subst7(<x1:var>,<x2:var>,<x3:var>,<x4:var>,<x5:var>,<x6:var>,<x7:var>.
                 <t:term:E>)
                 == is_subst(<x1>,<x2>,<x3>,<x4>,<x5>,<x6>,<x7>.<t>)
*D is_subst8   EdAlias is8 ::
                 is_subst8(<x1:var>,<x2:var>,<x3:var>,<x4:var>,<x5:var>,<x6:var>,<x7:var>,
                 <x8:var>.<t:term:E>)
                 == is_subst(<x1>,<x2>,<x3>,<x4>,<x5>,<x6>,<x7>,<x8>.<t>)
*R is_substFormation
               H ⊢ 𝕌 ext !call_ml{is_substFormation_extract_maker:s}

                 BY is_substFormation !call_ml

                    Let SubGoals ext = !call_ml{is_substFormation_rule:s}
                 SubGoals
*R is_substEquality
               H ⊢ t1 = t2 ∈ 𝕌 ext ext

                 BY is_substEquality !call_ml
```

```
                   Let SubGoals ext = !call_ml{is_substEquality_rule:s}
                 SubGoals
*C begin_term_eq             ---------- term_eq ----------
*D term_eq     EdAlias termeq ::
                 if <a:Term_term>=<b:Term_term> then <s:term:*> else <t:term:E>
                 == term_eq(<a>; <b>; <s>; <t>)
*R term_eqEquality
               H ⊢ if a1=b1 then s1 else t1 = if a2=b2 then s2 else t2 ∈ T

                 BY term_eqEquality v

                 H ⊢ a1 = a2 ∈ Term
                 H ⊢ b1 = b2 ∈ Term
                 H, v:(a1 = b1 ∈ Term) ⊢ s1 = s2 ∈ T
                 H, v:((a1 = b1 ∈ Term) ⟶ Void) ⊢ t1 = t2 ∈ T
*R term_eqReduceTrue
               H ⊢ if a=b then s else t = u ∈ T

                 BY term_eqReduceTrue ()

                 H ⊢ s = u ∈ T
                 H ⊢ a = b ∈ Term
*R term_eqReduceFalse
               H ⊢ if a=b then s else t = u ∈ T

                 BY term_eqReduceFalse ()

                 H ⊢ t = u ∈ T
                 H ⊢ (a = b ∈ Term) ⟶ Void
*M init_term_eq
               let term_eqEquality p =
                 (Refine ‘term_eqEquality‘ [mk_var_arg (new_invisible_var p)]
                  THENL
                  [Id; Id; Id;
                   OnLastHyp (\i. FoldAtAddr ‘false‘ [2] i
                                  THEN FoldTop ‘implies‘ i
                                  THEN FoldTop ‘not‘ i)]
                 ) p
               ;;

               update_EqCD_additions ‘term_eqEquality‘ term_eqEquality;
*A eq_term                   x =t y ==  if x=y then tt else ff
*D eq_term_df                <a:term:*> =t <b:term:*>== eq_term(<a>; <b>)
*T eq_term_wf               0 ∀x,y:Term.  x =t y ∈ 𝔹
*M eq_term_eval
               let eq_term_evalC = UnfoldTopC ‘eq_term‘ ANDTHENC RedexC
               ;;
*T decidable__term_equal    2 ∀s,t:Term.  Dec(s = t ∈ Term)
*C begin_TermAuto             ---------- TermAuto ----------
*R isTerm      H ⊢ t = t ∈ Term

                 BY isTerm ()

                 H ⊢ is_subst0(.t)
*R isSubstTerm H ⊢ is_subst0(.t)

                 BY isSubstTerm ()

                 H ⊢ t ∈ Term
*R isSubst     H ⊢ t ext ext
```

```
                BY isSubst ()

                   Let SubGoals ext = !call_ml{is_subst_rule:s}
                SubGoals
*R isTermSubst H ⊢ t ext ext

                BY isTermSubst !call_ml

                   Let SubGoals ext = !call_ml{isTermSubst_rule:s}
                SubGoals
*R isSubstThin H ⊢ t ext ext

                BY isSubstThin !call_ml

                   Let SubGoals ext = !call_ml{is_substThin_rule:s}
                SubGoals
*C begin_termin_termof        ---------- termin/termof ----------
*A termin                     x ↓∈ t ==  !null_abstraction
*D termin_df   Parens ::Prec(atomrel)::
                {->0}<x:term>{<-}{\\? }↓∈ {->0}<t:type>{<-}
                == termin(<t>; <x>)
*R terminEquality
              H ⊢ (t1 ↓∈ A1) = (t2 ↓∈ A2) ∈ 𝕌

                BY terminEquality ()

                H ⊢ A1 = A2 ∈ 𝕌
                H ⊢ t1 = t2 ∈ Term
*R terminFormation
              H ⊢ 𝕌 ext t ↓∈ A

                BY terminFormation ()

                H ⊢ 𝕌 ext A
                H ⊢ Term ext t
*T termin_wf                2 ∀a:Term. ∀A:𝕌.  (a ↓∈ A) ∈ 𝕌
*M init_termin let terminEquality = (Refine ‘terminEquality‘ []) ;;
              update_Trivial_additions ‘terminEquality‘ terminEquality ;;
*A termof                     Termof A ==  {t:Term| t ↓∈ A}
*D termof_df                  Parens ::Prec(postop)::  Termof <t:Type:E>== termof(<t>)
*T termof_wf                1 ∀A:𝕌. Termof A ∈ 𝕌
*C begin_up_down              ---------- up/down ----------
*D up                         Parens ::Prec(preop)::  ↑<t:term:E>== up(<t>)
*R upFormation H ⊢ Term ext ↑t

                BY upFormation ()

                H ⊢ Term ext t
*R upEquality  H ⊢ ↑t1 = ↑t2 ∈ Term

                BY upEquality ()

                H ⊢ t1 = t2 ∈ Term
*D down                       Parens ::Prec(preop)::  ↓<t:term:E>== down(<t>)
*R downFormation
              H ⊢ A ext ↓t

                BY downFormation ()
```

```
                H ⊢ Termof A ext t
*R downEquality
              H ⊢ ↓t1 = ↓t2 ∈ A

                BY downEquality ()

                H ⊢ t1 = t2 ∈ Termof A
*T up_down                  2 ∀t:Termof Term. ↑↓t = t ∈ Term
*T down_up                  1 ∀t:Term. ↓↑t = t ∈ Term
*T termin_member            0 ∀a:Term. ∀A:𝕌.  a ↓∈ A ⇒ ↓a ∈ A
*T termof_member            2 ∀A:𝕌. ∀a:Termof A.  ↓a ∈ A
*T up_termin                0 ∀x:Term. ↑x ↓∈ Term
*T up_wf                    2 ∀x:Term. ↑x ∈ Term
*T up_wf2                   2 ∀x:Term. ↑x ∈ Termof Term
*A term_downeq                t1 = t2 ↓∈ A ==  (t1 ↓∈ A ∧ t2 ↓∈ A) c∧ (↓t1 = ↓t2 ∈ A)
*D term_downeq_df
              Parens ::Prec(atomrel)::Index 0 ::
                {[SOFT]<t1:term:L>{\\ }= {->0}<t2:term:L>{<-}{\\ }↓∈ {->0}<A:term:L>{<-}{]}
                == term_downeq(<A>; <t1>; <t2>)
*T term_downeq_wf           3 ∀A:𝕌. ∀x1,x2:Termof A.  (x1 = x2 ↓∈ A) ∈ ℙ
*C begin_reps                 ---------- reps ----------
*A reps                       x ↓= y ∈ t ==  (x ↓∈ t) c∧ (↓x = y ∈ t)
*D reps_df     Parens ::Prec(atomrel)::Index 0 ::
                {[SOFT]<x:term:L>{\\ }↓= {->0}<y:term:L>{<-}{\\ }∈ {->0}<t:term:L>{<-}{]}
                == reps(<t>; <x>; <y>)
*T reps_wf                  2 ∀u:𝕌. ∀t:Termof u. ∀x:u.  (t ↓= x ∈ u) ∈ ℙ
*A repst                      x ↓= y ==  x ↓= y ∈ Term
*D repst_df    EdAlias repsterm ::Parens ::Prec(atomrel)::Index 0 ::
                {[SOFT]<x:term:L>{\\ }↓= {->0}<y:term:L>{<-}{]}
                == repst(<x>; <y>)
*T reps_wf2                 2 ∀t,x:Term.  (t ↓= x ∈ Term) ∈ ℙ
*T repst_wf                 2 ∀t,x:Term.  (t ↓= x) ∈ ℙ
*T up_reps                  2 ∀t:Term. ↑t ↓= t ∈ Term
*T up_repst                 2 ∀t:Term. ↑t ↓= t
*C begin_term_subst           ---------- term_subst ----------
*D term_subst  EdAlias termsubst ::Parens ::Prec(atomrel)::Index 0 ::
                {[SOFT]<t:term:L>[{->0}<s:term:L>/<x:term:L>{<-}]
                == term_subst(<t>; <x>; <s>)
*T term_subst_wf            0 ∀t1,t2,t3:Term.  (t1[t2/t3]) ∈ Term
*A term_closed                term_closed(t) ==  ∀s,v:Term.  (t[s/v]) = t ∈ Term
*T term_closed_wf           2 ∀t:Term. term_closed(t) ∈ ℙ
*T term_term_closed         0 ∀t:Termof Term. term_closed(t)
*T up_term_closed           1 ∀t:Term. term_closed((↑t))
*A free_in                    free_in(v; t) ==  ¬(∀s:Term. (t[s/v]) = t ∈ Term)
*T free_in_wf               2 ∀t,v:Term.  free_in(v; t) ∈ ℙ
#T free_iff_not_closed      3 ∀t:Term. (∃v:Term. free_in(v; t)) ⟺ ¬term_closed(t)
*C begin_utilities            ---------- utilities ----------
*D idform_color_df
              {BAD-COLOR(<type:type>,<c:color>)}== !dform_color{<type>:t, <c>:n}
              C<c:color-num>{== !dform_color{B:t, <c>:n}
              C+<c:color-num>{== !dform_color{B+:t, <c>:n}
              C-<c:color-num>{== !dform_color{B-:t, <c>:n}
              }<c:color-num>C== !dform_color{E:t, <c>:n}
              Q<c:quotedness>{== !dform_color{BQ:t, <c>:n}
              }<c:color-num>Q== !dform_color{EQ:t, <c>:n}
*D hypertext   EdAlias htext ::(HyperLink)::  C10{<<t:text:*>>}10C== !hyperlink{<t>:s}
*D hyperterm   EdAlias hterm ::(HyperLink)::  C10{⌜<t:term:*>⌝}10C== !hyperlink(<t>)
*C reflection_end             **********************************
```

```
============================== Theorems: Full Printout ==============================

<<< TERM_WF >>>
⊢ Term ∈ 𝕌
|
BY Unfold ‘member‘ 0
|
⊢ Term = Term ∈ 𝕌
|
BY TermEquality

<<< EQ_TERM_WF >>>
⊢ ∀x,y:Term.  x =t y ∈ 𝔹
|
BY Unfold ‘eq_term‘ 0 THEN Auto

<<< DECIDABLE__TERM_EQUAL >>>
⊢ ∀s,t:Term.  Dec(s = t ∈ Term)
|
BY Unfold ‘decidable‘ 0 THEN Auto
|
1. s: Term
2. t: Term
⊢ s = t ∈ Term ∨ ¬(s = t ∈ Term)
|
BY UseWitness ⌈if s=t then inl Ax  else (inr (λx.x) )⌉
THEN Auto
|
3. ¬(s = t ∈ Term)
⊢ (λx.x) ∈ ¬(s = t ∈ Term)
|
BY Unfold ‘not‘ 0 THEN Auto

<<< TERMIN_WF >>>
⊢ ∀a:Term. ∀A:𝕌.  (a ↓∈ A) ∈ 𝕌
|
BY GenUnivCD
|\
| 1. a: Term
| 2. A: 𝕌
| ⊢ (a ↓∈ A) ∈ 𝕌
| |
1 BY Unfold ‘member‘ 0
| |
| ⊢ (a ↓∈ A) = (a ↓∈ A) ∈ 𝕌
| |
1 BY Auto
|\
| 1. a: Term
| ⊢ 𝕌 ∈ 𝕌’
| |
1 BY Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< TERMOF_WF >>>
```

```
⊢ ∀A:𝕌. Termof A ∈ 𝕌
|
BY GenUnivCD
|\
| 1. A: 𝕌
| ⊢ Termof A ∈ 𝕌
| |
1 BY Unfold ‘termof‘ 0 THEN Auto
 \
  ⊢ 𝕌 ∈ 𝕌’
  |
  BY Auto

<<< UP_DOWN >>>
⊢ ∀t:Termof Term. ↑↓t = t ∈ Term
|
BY GenUnivCD
|\
| 1. t: Termof Term
| ⊢ ↑↓t = t ∈ Term
| |
1 BY ComputeOpid ‘up‘ 0
| |
| ⊢ t = t ∈ Term
| |
1 BY D 1 THEN Auto
 \
  ⊢ Termof Term ∈ 𝕌
  |
  BY Auto

<<< DOWN_UP >>>
⊢ ∀t:Term. ↓↑t = t ∈ Term
|
BY GenUnivCD
|\
| 1. t: Term
| ⊢ ↓↑t = t ∈ Term
| |
1 BY ComputeOpid ‘down‘ 0 THEN Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< TERMIN_MEMBER >>>
⊢ ∀a:Term. ∀A:𝕌.  a ↓∈ A ⇒ ↓a ∈ A
|
BY Fiat

<<< TERMOF_MEMBER >>>
⊢ ∀A:𝕌. ∀a:Termof A.  ↓a ∈ A
|
BY GenUnivCD THENA Auto
|
1. A: 𝕌
2. a: Termof A
⊢ ↓a ∈ A
```

```
|
BY D 2
|
2. a: Term
3. a ↓∈ A
|
BY BLemma ‘termin_member‘
THEN Auto

<<< UP_TERMIN >>>
⊢ ∀x:Term. ↑x ↓∈ Term
|
BY Fiat

<<< UP_WF >>>
⊢ ∀x:Term. ↑x ∈ Term
|
BY GenUnivCD
|\
| 1. x: Term
| ⊢ ↑x ∈ Term
| |
1 BY Unfold ‘member‘ 0
THEN Refine ‘upEquality‘ []
THEN Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< UP_WF2 >>>
⊢ ∀x:Term. ↑x ∈ Termof Term
|
BY GenUnivCD
|\
| 1. x: Term
| ⊢ ↑x ∈ Termof Term
| |
1 BY MemTypeCD THENA Auto
| |\
| | ⊢ ↑x ∈ Term
| | |
1 2 BY Auto
|  \
|   ⊢ ↑x ↓∈ Term
|   |
1   BY BLemma ‘up_termin‘ THENA Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< TERM_DOWNEQ_WF >>>
⊢ ∀A:𝕌. ∀x1,x2:Termof A.  (x1 = x2 ↓∈ A) ∈ ℙ
|
BY GenUnivCD
|\
| 1. A: 𝕌
```

```
| 2. x1: Termof A
| 3. x2: Termof A
| ⊢ (x1 = x2 ↓∈ A) ∈ ℙ
| |
1 BY Unfold ‘term_downeq‘ 0
| |
| ⊢ (x1 ↓∈ A ∧ x2 ↓∈ A) c∧ (↓x1 = ↓x2 ∈ A) ∈ ℙ
| |
1 BY OnHyps [2;3] (Unfold ‘termof‘) THEN Auto
| |\
| | 2. x1: {t:Term| t ↓∈ A}
| | 3. x2: {t:Term| t ↓∈ A}
| | 4. x1 ↓∈ A
| | 5. x2 ↓∈ A
| | ⊢ ↓x1 ∈ A
| | |
1 2 BY BLemma ‘termin_member‘ THEN Auto
|  \
|   2. x1: {t:Term| t ↓∈ A}
|   3. x2: {t:Term| t ↓∈ A}
|   4. x1 ↓∈ A
|   5. x2 ↓∈ A
|   ⊢ ↓x2 ∈ A
|   |
1   BY BLemma ‘termin_member‘ THEN Auto
|\
| 1. A: 𝕌
| 2. x1: Termof A
| ⊢ Termof A ∈ 𝕌
| |
1 BY Auto
|\
| 1. A: 𝕌
| ⊢ Termof A ∈ 𝕌
| |
1 BY Auto
 \
  ⊢ 𝕌 ∈ 𝕌'
  |
  BY Auto

<<< REPS_WF >>>
⊢ ∀u:𝕌. ∀t:Termof u. ∀x:u.  (t ↓= x ∈ u) ∈ ℙ
|
BY GenUnivCD
|\
| 1. u: 𝕌
| 2. t: Termof u
| 3. x: u
| ⊢ (t ↓= x ∈ u) ∈ ℙ
| |
1 BY Unfold ‘reps‘ 0
| |
| ⊢ (t ↓∈ u) c∧ (↓t = x ∈ u) ∈ ℙ
| |
1 BY D 2 THEN Auto
| |
```

```
| 2. t: Term
| 3. t ↓∈ u
| 4. x: u
| 5. t ↓∈ u
| ⊢ ↓t ∈ u
| |
1 BY BLemma ‘termin_member‘ THEN Auto
|\
| 1. u: 𝕌
| 2. t: Termof u
| ⊢ u ∈ 𝕌
| |
1 BY Auto
|\
| 1. u: 𝕌
| ⊢ Termof u ∈ 𝕌
| |
1 BY Auto
 \
  ⊢ 𝕌 ∈ 𝕌'
  |
  BY Auto

<<< REPS_WF2 >>>
⊢ ∀t,x:Term.  (t ↓= x ∈ Term) ∈ ℙ
|
BY GenUnivCD
|\
| 1. t: Term
| 2. x: Term
| ⊢ (t ↓= x ∈ Term) ∈ ℙ
| |
1 BY Unfold ‘reps‘ 0
| |
| ⊢ (t ↓∈ Term) c∧ (↓t = x ∈ Term) ∈ ℙ
| |
1 BY Auto
| |
| 3. t ↓∈ Term
| ⊢ is_subst0(.(↓t))
| |
1 BY FLemma ‘termin_member‘ [3]
THEN Auto
|\
| 1. t: Term
| ⊢ Term ∈ 𝕌
| |
1 BY Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< REPST_WF >>>
⊢ ∀t,x:Term.  (t ↓= x) ∈ ℙ
|
BY GenUnivCD
|\
```

```
| 1. t: Term
| 2. x: Term
| ⊢ (t ↓= x) ∈ ℙ
| |
1 BY Unfold ‘repst‘ 0
| |
| ⊢ (t ↓= x ∈ Term) ∈ ℙ
| |
1 BY BLemma ‘reps_wf2‘ THEN Auto
|\
| 1. t: Term
| ⊢ Term ∈ 𝕌
| |
1 BY Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< UP_REPS >>>
⊢ ∀t:Term. ↑t ↓= t ∈ Term
|
BY GenUnivCD THENA Auto
|
1. t: Term
⊢ ↑t ↓= t ∈ Term
|
BY Unfold ‘reps‘ 0
|
⊢ (↑t ↓∈ Term) c∧ (↓↑t = t ∈ Term)
|
BY D 0
|\
| ⊢ ↑t ↓∈ Term
| |
1 BY BLemma ‘up_termin‘ THENA Auto
 \
  2. ↑t ↓∈ Term
  ⊢ ↓↑t = t ∈ Term
  |
  BY BLemma ‘down_up‘ THENA Auto

<<< UP_REPST >>>
⊢ ∀t:Term. ↑t ↓= t
|
BY GenUnivCD THENA Auto
|
1. t: Term
⊢ ↑t ↓= t
|
BY Unfold ‘repst‘ 0
THEN BLemma ‘up_reps‘
THEN Auto

<<< TERM_SUBST_WF >>>
⊢ ∀t1,t2,t3:Term.  (t1[t2/t3]) ∈ Term
|
BY Fiat
```

```
<<< TERM_CLOSED_WF >>>
⊢ ∀t:Term. term_closed(t) ∈ ℙ
|
BY GenUnivCD
|\
| 1. t: Term
| ⊢ term_closed(t) ∈ ℙ
| |
1 BY Unfold ‘term_closed‘ 0
THEN Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< TERM_TERM_CLOSED >>>
⊢ ∀t:Termof Term. term_closed(t)
|
BY Fiat

<<< UP_TERM_CLOSED >>>
⊢ ∀t:Term. term_closed((↑t))
|
BY GenUnivCD THENA Auto
|
1. t: Term
⊢ term_closed((↑t))
|
BY BLemma ‘term_term_closed‘
|
⊢ ↑t ∈ Termof Term
|
BY Auto

<<< FREE_IN_WF >>>
⊢ ∀t,v:Term.  free_in(v; t) ∈ ℙ
|
BY GenUnivCD
|\
| 1. t: Term
| 2. v: Term
| ⊢ free_in(v; t) ∈ ℙ
| |
1 BY Unfold ‘free_in‘ 0 THEN Auto
|\
| 1. t: Term
| ⊢ Term ∈ 𝕌
| |
1 BY Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< FREE_IFF_NOT_CLOSED >>> (partial)
⊢ ∀t:Term. (∃v:Term. free_in(v; t)) ⟺ ¬term_closed(t)
|
BY D 0 THENA Auto
```

```
|
1. t: Term
⊢ (∃v:Term. free_in(v; t)) ⟺ ¬term_closed(t)
|
BY D 0
|\
| ⊢ (∃v:Term. free_in(v; t)) ⇒ ¬term_closed(t)
| |
1 BY D 0 THENA Auto
| |
| 2. ∃v:Term. free_in(v; t)
| ⊢ ¬term_closed(t)
| |
1 BY D 0 THENA Auto
| |
| 3. term_closed(t)
| ⊢ False
| |
1 BY Unfold ‘term_closed‘ 3
| |
| 3. ∀s,v:Term.  (t[s/v]) = t ∈ Term
| |
1 BY D 2
| |
| 2. v: Term
| 3. free_in(v; t)
| 4. ∀s,v:Term.  (t[s/v]) = t ∈ Term
| |
1 BY Unfold ‘free_in‘ 3
| |
| 3. ¬(∀s:Term. (t[s/v]) = t ∈ Term)
| |
1 BY D 3
| |
| 3. ∀s,v:Term.  (t[s/v]) = t ∈ Term
| ⊢ ∀s:Term. (t[s/v]) = t ∈ Term
| |
1 BY D 0 THENA Auto
| |
| 4. s: Term
| ⊢ (t[s/v]) = t ∈ Term
| |
1 BY With ⌜s⌝ (D 3) THENA Auto
| |
| 3. s: Term
| 4. ∀v:Term. (t[s/v]) = t ∈ Term
| |
1 BY With ⌜v⌝ (D 4) THENA Auto
| |
| 4. (t[s/v]) = t ∈ Term
| |
1 BY Auto
 \
  ⊢ (∃v:Term. free_in(v; t)) ⇐ ¬term_closed(t)
  |
  BY D 0 THENA Auto
  |
```

```
  2. ¬term_closed(t)
  ⊢ ∃v:Term. free_in(v; t)
  |
  BY % To do this we need to know that the goal is decidable
  and this is only possible with induction. %
Id
  |
  |
  INCOMPLETE
```