TARSKI

---

*C tarski_begin ********** TARSKI **********

*C TarskiText We're assuming we have the type of terms (⌜Term⌝) and a representation relation between terms (⌜· ↓= ·⌝).

a. We assume that if ⌜t ↓= s⌝ then ⌜t⌝ is closed.
<up_term_closed>

Notation and some simple corrolaries (indicated by "thus"):
(There are also assumptions about substitution into ⌜subx(·; ·)⌝ and ⌜↑·⌝)

a1. ⌜x⌝ is a variable
a2. ⌜subx(t; e)⌝ is substitution of term ⌜e⌝ for variable ⌜x⌝ in ⌜t⌝
a3. ⌜t ↓= t' ∧ r ↓= r' ⇒ subx(t; r) ↓= subx(t'; r')⌝
<qsubx_repst>
a4. ⌜subx(subx(t; r); e) = subx(subx(t; e); subx(r; e)) ∈ Term⌝
<qsubx_subx>
a5. ⌜↑t ↓= t⌝
<up_repst>
a6. ⌜t ↓= r ⇒ ↑t ↓= ↑r⌝
<qup_repst>
(note that ⌜t ↓= r ⇒ ↑↑t ↓= ↑r⌝ does not work! <up_up_repst> -- Eli)

b. Thus, ⌜↑↑t ↓= ↑t⌝
<qup_up_repst>
b1. ⌜subx(↑t; e) = ↑subx(t; e) ∈ Term⌝
<qup_subx>

c1. ⌜f(t)⌝ is ⌜subx((↑t); subx(x; (↑x)))⌝
c2. ⌜s(t)⌝ is ⌜subx(f(t); (↑f(t)))⌝

c3. Thus, ⌜s(t) = subx((↑t); subx((↑f(t)); (↑↑f(t)))) ∈ Term⌝ by (a) on ⌜↑t⌝
<s1>
c4. Thus, ⌜s(t) ↓= subx(t; subx(f(t); (↑f(t))))⌝ by (b)
<s2>
c. Thus, ⌜s(t) ↓= subx(t; s(t))⌝
<s_reps>

d1. ⌜¬t⌝ is the term built from term ⌜t⌝ by the negation-denoting operator
d. Thus ⌜subx(¬t; e) = ¬subx(t; e) ∈ Term⌝
<qnot_subx>

The Tarskian argument:

Let ⌜RepsTruth(L; Tr; tr)⌝ where ⌜L⌝ and ⌜Tr⌝ are properties of terms and ⌜tr⌝ is a term, mean:
1. ⌜∀S:Term. (∃t:Term. S ↓= t) ⇒ L subx(tr; S)⌝
2. ⌜RespectsNot(Tr; L)⌝
meaning: ⌜∀t:Term. Tr ¬t ⟺ L t ∧ ¬(Tr t)⌝
3. ⌜ReflectsProp(Tr; tr; Tr)⌝
meaning: ⌜∀t,qt:Term. qt ↓= t ⇒ {Tr subx(tr; qt) ⟺ Tr t}⌝

This is meant to be part of the criterion for ⌜Tr⌝ being a truth predicate on ⌜L⌝, and for ⌜tr⌝ to denote ⌜Tr⌝ (in ⌜x⌝).

```
                <Tarski>
                Then there are no ⌜L⌝, ⌜Tr⌝, ⌜tr⌝ such that ⌜RepsTruth(L; Tr; tr)⌝ thus:
                4.  Assume ⌜RepsTruth(L; Tr; tr)⌝
                    {Tarski:t}
                5.  let ⌜S = s((¬tr)) ∈ Term⌝
                    {Tarski:t11}
                6.  ⌜S ↓= ¬subx(tr; S)⌝ by (5,c,d)
                    {Tarski:t111}
                7.  ⌜L subx(tr; S)⌝ by (4,1,6)
                    {Tarski:t1112}
                8.  ⌜Tr subx(tr; S) ⟺ Tr ¬subx(tr; S)⌝ by (4,3,6)
                    {Tarski:t11121}
                9.  ⌜Tr ¬subx(tr; S) ⟺ L subx(tr; S) ∧ ¬(Tr subx(tr; S))⌝ by (4,2)
                    {Tarski:t111211}
                10. ⌜Tr ¬subx(tr; S) ⟺ ¬(Tr subx(tr; S))⌝ by (9,7)
                    {Tarski:t1112111}
                *.  ⌜Tr subx(tr; S) ⟺ ¬(Tr subx(tr; S))⌝ by (8,10)
                    {Tarski:t11121111}
                    ... which is false so (4) is false.
*T qup_termin               0 ∀t:Term. t ↓∈ Term ⇒ ↑t ↓∈ Term
*T qsubx_termin             0 ∀t,r:Term.  t ↓∈ Term ∧ r ↓∈ Term ⇒ subx(t; r) ↓∈ Term
*T push_down_qup            2 ∀t:Term. t ↓∈ Term ⇒ ↓↑t = ↑↓t ∈ Term
*T push_down_qsubx          2
                ∀t,r:Term.
                  subx(t; r) ↓∈ Term
                  ⇒ t ↓∈ Term
                  ⇒ r ↓∈ Term
                  ⇒ ↓subx(t; r) = subx((↓t); (↓r)) ∈ Term
#T up_up_repst              3 ∀t,r:Term.  t ↓= r ⇒ ↑↑t ↓= ↑r
*T qup_repst                2 ∀t,r:Term.  t ↓= r ⇒ ↑t ↓= ↑r
*T qup_up_repst             1 ∀t:Term. ↑↑t ↓= ↑t
*A subx                       subx(t; e) ==  t[e/x]
*T subx_wf                  2 ∀t,r:Term.  subx(t; r) ∈ Term
*T qsubx_repst              3 ∀t,r,t',r':Term.  t ↓= t' ∧ r ↓= r' ⇒ subx(t; r) ↓= subx(t'; r')
*T qsubx_subx               1
                ∀t,r,e:Term.  subx(subx(t; r); e) = subx(subx(t; e); subx(r; e)) ∈ Term
*T qup_subx                 1 ∀t,e:Term.  subx((↑t); e) = ↑subx(t; e) ∈ Term
*T qnot_subx                1 ∀t,e:Term.  subx((¬t); e) = ¬subx(t; e) ∈ Term
*A f                          f(t) ==  subx((↑t); subx(x; (↑x)))
*T f_wf                     1 ∀t:Term. f(t) ∈ Term
*A s                          s(t) ==  subx(f(t); (↑f(t)))
*T s_wf                     1 ∀t:Term. s(t) ∈ Term
*T s1                       3 ∀t:Term. s(t) = subx((↑t); subx((↑f(t)); (↑↑f(t)))) ∈ Term
*T s2                       3 ∀t:Term. s(t) ↓= subx(t; subx(f(t); (↑f(t))))
*T s_reps                   2 ∀t:Term. s(t) ↓= subx(t; s(t))
*A RespectsNot                RespectsNot(Tr; L) ==  ∀t:Term. Tr ¬t ⟺ L t ∧ ¬(Tr t)
*T RespectsNot_wf           2 ∀Tr,L:Term → ℙ.  RespectsNot(Tr; L) ∈ ℙ
*A ReflectsProp
                ReflectsProp(P; qP; Tr) ==  ∀t,qt:Term.  qt ↓= t ⇒ {Tr subx(qP; qt) ⟺ P t}
*T ReflectsProp_wf          2 ∀P:Term → ℙ. ∀qP:Term. ∀L:Term → ℙ.  ReflectsProp(P; qP; L) ∈ ℙ
*A RepsTruth   RepsTruth(L; Tr; tr) ==
                  (∀S:Term. (∃t:Term. S ↓= t) ⇒ L subx(tr; S))
                  ∧ RespectsNot(Tr; L)
                  ∧ ReflectsProp(Tr; tr; Tr)
*T RepsTruth_wf             2 ∀Tr:Term → ℙ. ∀tr:Term. ∀L:Term → ℙ.  RepsTruth(L; Tr; tr) ∈ ℙ
*T prop_and_iff             0 ∀A,B,C:ℙ.  B ⇒ (A ⟺ B ∧ C) ⇒ {A ⟺ C}
*T prop_iff_trans           0 ∀A,B,C:ℙ.  (A ⟺ B) ⇒ (B ⟺ C) ⇒ {A ⟺ C}
```

```
*T prop_iff_contra          0 ∀P:ℙ. (P ⟺ ¬P) ⇒ False
*T Tarski                   4 ¬(∃Tr:Term → ℙ. ∃tr:Term. ∃L:Term → ℙ. RepsTruth(L; Tr; tr))
*C tarski_end                 ********************************

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

<<< QUP_TERMIN >>>
⊢ ∀t:Term. t ↓∈ Term ⇒ ↑̲t ↓∈ Term
|
BY Fiat

<<< QSUBX_TERMIN >>>
⊢ ∀t,r:Term.  t ↓∈ Term ∧ r ↓∈ Term ⇒ subx(t; r) ↓∈ Term
|
BY Fiat

<<< PUSH_DOWN_QUP >>>
⊢ ∀t:Term. t ↓∈ Term ⇒ ↓↑̲t = ↑↓t ∈ Term
|
BY GenUnivCD THENA Auto
|
1. t: Term
2. t ↓∈ Term
⊢ ↓↑̲t = ↑↓t ∈ Term
|
BY ComputeOpid ‘down‘ 0
|
⊢ ↑↓t = ↑↓t ∈ Term
|
BY RWHL ‘up_down‘ 0
THEN Auto
THEN Unfold ‘termof‘ 0
THEN Auto

<<< PUSH_DOWN_QSUBX >>>
⊢ ∀t,r:Term.
  subx(t; r) ↓∈ Term ⇒ t ↓∈ Term ⇒ r ↓∈ Term ⇒ ↓subx(t; r) = subx((↓t); (↓r)) ∈ Term
|
BY GenUnivCD THENA Auto
|
1. t: Term
2. r: Term
3. subx(t; r) ↓∈ Term
4. t ↓∈ Term
5. r ↓∈ Term
⊢ ↓subx(t; r) = subx((↓t); (↓r)) ∈ Term
|
BY ComputeOpid ‘down‘ 0
|
⊢ subx((↓t); (↓r)) = subx((↓t); (↓r)) ∈ Term
|
BY Auto
|\
| ⊢ is_subst0(.(↓t))
| |
1 BY FLemma ‘termin_member‘ [4]
THEN Auto
 \
```

```
  ⊢ is_subst0(.(↓r))
  |
  BY FLemma ‘termin_member‘ [5]
THEN Auto

<<< UP_UP_REPST >>> (partial)
⊢ ∀t,r:Term.  t ↓= r ⇒ ↑↑t ↓= ↑r
|
BY GenUnivCD THENA Auto
|
1. t: Term
2. r: Term
3. t ↓= r
⊢ ↑↑t ↓= ↑r
|
BY OnHyps [3;0] (Unfold ‘repst‘)
|
3. t ↓= r ∈ Term
⊢ ↑↑t ↓= ↑r ∈ Term
|
BY OnHyps [3;0] (Unfold ‘reps‘)
|
3. (t ↓∈ Term) c∧ (↓t = r ∈ Term)
⊢ (↑↑t ↓∈ Term) c∧ (↓↑↑t = ↑r ∈ Term)
|
BY D 3 THEN D 0
|\
| 3. t ↓∈ Term
| 4. ↓t = r ∈ Term
| ⊢ ↑↑t ↓∈ Term
| |
1 BY BLemma ‘up_termin‘ THEN Auto
 \
  3. t ↓∈ Term
  4. ↓t = r ∈ Term
  5. ↑↑t ↓∈ Term
  ⊢ ↓↑↑t = ↑r ∈ Term
  |
  BY RWHL ‘down_up‘ 0 THENA Auto
  |
  ⊢ ↑t = ↑r ∈ Term
  |
  BY Thin 5
  |
  |
  BY % Obviously bogus! % Id
  |
  |
  INCOMPLETE

<<< QUP_REPST >>>
⊢ ∀t,r:Term.  t ↓= r ⇒ ⇑t ↓= ↑r
|
BY GenUnivCD THENA Auto
|
1. t: Term
2. r: Term
3. t ↓= r
```

```
⊢ ↑̲t ↓= ↑r
|
BY OnHyps [0;3] (Unfold ‘repst‘)
|
3. t ↓= r ∈ Term
⊢ ↑̲t ↓= ↑r ∈ Term
|
BY OnHyps [0;3] (Unfold ‘reps‘)
|
3. (t ↓∈ Term) c∧ (↓t = r ∈ Term)
⊢ (↑̲t ↓∈ Term) c∧ (↓↑̲t = ↑r ∈ Term)
|
BY D 3 THEN D 0
|\
| 3. t ↓∈ Term
| 4. ↓t = r ∈ Term
| ⊢ ↑̲t ↓∈ Term
| |
1 BY BLemma ‘qup_termin‘ THEN Auto
 \
  3. t ↓∈ Term
  4. ↓t = r ∈ Term
  5. ↑̲t ↓∈ Term
  ⊢ ↓↑̲t = ↑r ∈ Term
  |
  BY RWHL ‘push_down_qup‘ 0 THENA Auto
  |
  ⊢ ↑↓t = ↑r ∈ Term
  |
  BY HypSubst 4 0 THEN Auto

<<< QUP_UP_REPST >>>
⊢ ∀t:Term. ↑̲↑t ↓= ↑t
|
BY GenUnivCD THENA Auto
|
1. t: Term
⊢ ↑̲↑t ↓= ↑t
|
BY BLemma ‘qup_repst‘ THENA Auto
|
⊢ ↑t ↓= t
|
BY BLemma ‘up_repst‘ THENA Auto

<<< SUBX_WF >>>
⊢ ∀t,r:Term.  subx(t; r) ∈ Term
|
BY GenUnivCD
|\
| 1. t: Term
| 2. r: Term
| ⊢ subx(t; r) ∈ Term
| |
1 BY Unfold ‘subx‘ 0 THEN Auto
|\
| 1. t: Term
| ⊢ Term ∈ 𝕌
```

```
| |
1 BY Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto
```

<<< QSUBX_REPST >>>

```
⊢ ∀t,r,t',r':Term.  t ↓= t' ∧ r ↓= r' ⇒ subx(t; r) ↓= subx(t'; r')
|
BY GenUnivCD THEN Auto
|
1. t: Term
2. r: Term
3. t': Term
4. r': Term
5. t ↓= t'
6. r ↓= r'
⊢ subx(t; r) ↓= subx(t'; r')
|
BY OnHyps [0;5;6] (Unfold `repst`)
|
5. t ↓= t' ∈ Term
6. r ↓= r' ∈ Term
⊢ subx(t; r) ↓= subx(t'; r') ∈ Term
|
BY OnHyps [0;5;6] (Unfold `reps`)
|
5. (t ↓∈ Term) c∧ (↓t = t' ∈ Term)
6. (r ↓∈ Term) c∧ (↓r = r' ∈ Term)
⊢ (subx(t; r) ↓∈ Term) c∧ (↓subx(t; r) = subx(t'; r') ∈ Term)
|
BY OnHyps [6;5;0] D
|\
| 5. t ↓∈ Term
| 6. ↓t = t' ∈ Term
| 7. r ↓∈ Term
| 8. ↓r = r' ∈ Term
| ⊢ subx(t; r) ↓∈ Term
| |
1 BY BLemma `qsubx_termin` THEN Auto
 \
  5. t ↓∈ Term
  6. ↓t = t' ∈ Term
  7. r ↓∈ Term
  8. ↓r = r' ∈ Term
  9. subx(t; r) ↓∈ Term
  ⊢ ↓subx(t; r) = subx(t'; r') ∈ Term
  |
  BY RWHL `push_down_qsubx` 0
THENA Auto
  |
  ⊢ subx((↓t); (↓r)) = subx(t'; r') ∈ Term
  |
  BY HypSubst 6 0 THEN HypSubst 8 0
THEN Auto
```

<<< QSUBX_SUBX >>>

```
⊢ ∀t,r,e:Term.  subx(subx(t; r); e) = subx(subx(t; e); subx(r; e)) ∈ Term
|
BY GenUnivCD THENA Auto
|
1. t: Term
2. r: Term
3. e: Term
⊢ subx(subx(t; r); e) = subx(subx(t; e); subx(r; e)) ∈ Term
|
BY Unfold ‘subx‘ 0
|
⊢ (subx(t; r)[e/x]) = subx((t[e/x]); (r[e/x])) ∈ Term
|
BY ComputeAtAddr [2] 0 THEN Auto

<<< QUP_SUBX >>>
⊢ ∀t,e:Term.  subx((↑t); e) = ↑subx(t; e) ∈ Term
|
BY GenUnivCD THENA Auto
|
1. t: Term
2. e: Term
⊢ subx((↑t); e) = ↑subx(t; e) ∈ Term
|
BY Unfold ‘subx‘ 0
|
⊢ (↑t[e/x]) = ↑(t[e/x]) ∈ Term
|
BY ComputeAtAddr [2] 0 THEN Auto

<<< QNOT_SUBX >>>
⊢ ∀t,e:Term.  subx((¬t); e) = ¬subx(t; e) ∈ Term
|
BY GenUnivCD THENA Auto
|
1. t: Term
2. e: Term
⊢ subx((¬t); e) = ¬subx(t; e) ∈ Term
|
BY Unfold ‘subx‘ 0
|
⊢ (¬t[e/x]) = ¬(t[e/x]) ∈ Term
|
BY ComputeAtAddr [2] 0 THEN Auto

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

```
<<< S_WF >>>
⊢ ∀t:Term. s(t) ∈ Term
|
BY GenUnivCD
|\
| 1. t: Term
| ⊢ s(t) ∈ Term
| |
1 BY Unfold `s` 0 THEN Auto
 \
  ⊢ Term ∈ 𝕌
  |
  BY Auto

<<< S1 >>>
⊢ ∀t:Term. s(t) = subx((↑t); subx((↑f(t)); (↑↑f(t)))) ∈ Term
|
BY GenUnivCD THENA Auto
|
1. t: Term
⊢ s(t) = subx((↑t); subx((↑f(t)); (↑↑f(t)))) ∈ Term
|
BY Assert ⌜s(t) = subx(f(t); (↑f(t))) ∈ Term⌝
|\
| ⊢ s(t) = subx(f(t); (↑f(t))) ∈ Term
| |
1 BY Unfold `s` 0 THEN Auto
 \
  2. s(t) = subx(f(t); (↑f(t))) ∈ Term
  |
  BY Unfold `subx` 2
  |
  2. s(t) = (f(t)[↑f(t)/x]) ∈ Term
  |
  BY ComputeWithTaggedTerm
  ⌜s(t) = ([0:f(t)][↑f(t)/x]) ∈ Term⌝
  2
  |
  2. s(t) = (subx((↑t); subx(x; (↑x)))[↑f(t)/x]) ∈ Term
  |
  BY Repeat (ComputeOpid `term_subst` 2)
  |
  2. s(t) = subx((↑t[↑f(t)/x]); subx((↑f(t)); (↑↑f(t)))) ∈ Term
  |
  BY Assert ⌜(↑t[↑f(t)/x]) = ↑t ∈ Term⌝
  |\
  | ⊢ (↑t[↑f(t)/x]) = ↑t ∈ Term
  | |
  1 BY BLemmaWithUnfolds ``term_closed`` `up_term_closed`
THEN Auto
   \
    3. (↑t[↑f(t)/x]) = ↑t ∈ Term
    |
    BY HypSubst 3 2 THEN Auto

<<< S2 >>>
⊢ ∀t:Term. s(t) ↓= subx(t; subx(f(t); (↑f(t))))
|
```

```
BY GenUnivCD THENA Auto
|
1. t: Term
⊢ s(t) ↓= subx(t; subx(f(t); (↑f(t))))
|
BY RWHL ‘s1‘ 0 THENA Auto
|
⊢ subx((↑t); subx((↑f(t)); (↑↑f(t)))) ↓= subx(t; subx(f(t); (↑f(t))))
|
BY BLemma ‘qsubx_repst‘ THEN Auto
|\
| ⊢ ↑t ↓= t
| |
1 BY BLemma ‘up_repst‘ THEN Auto
 \
  ⊢ subx((↑f(t)); (↑↑f(t))) ↓= subx(f(t); (↑f(t)))
  |
  BY BLemma ‘qsubx_repst‘ THENA Auto
  |
  ⊢ ↑f(t) ↓= f(t) ∧ ↑↑f(t) ↓= ↑f(t)
  |
  BY D 0
  |\
  | ⊢ ↑f(t) ↓= f(t)
  | |
  1 BY BLemma ‘up_repst‘ THEN Auto
   \
    ⊢ ↑↑f(t) ↓= ↑f(t)
    |
    BY BLemma ‘qup_up_repst‘ THEN Auto

<<< S_REPS >>>
⊢ ∀t:Term. s(t) ↓= subx(t; s(t))
|
BY GenUnivCD THENA Auto
|
1. t: Term
⊢ s(t) ↓= subx(t; s(t))
|
BY ComputeWithTaggedTerm ⌜s(t) ↓= subx(t; [1:s(t)])⌝ 0
THENA Auto
|
⊢ s(t) ↓= subx(t; subx(f(t); (↑f(t))))
|
BY BLemma ‘s2‘ THEN Auto

<<< RESPECTSNOT_WF >>>
⊢ ∀Tr,L:Term → ℙ.  RespectsNot(Tr; L) ∈ ℙ
|
BY GenUnivCD
|\
| 1. Tr: Term → ℙ
| 2. L: Term → ℙ
| ⊢ RespectsNot(Tr; L) ∈ ℙ
| |
1 BY Unfold ‘RespectsNot‘ 0 THEN Auto
|\
| 1. Tr: Term → ℙ
```

```
| ⊢ Term → ℙ ∈ 𝕌'
| |
1 BY Auto
 \
  ⊢ Term → ℙ ∈ 𝕌'
  |
  BY Auto

<<< REFLECTSPROP_WF >>>
⊢ ∀P:Term → ℙ. ∀qP:Term. ∀L:Term → ℙ.  ReflectsProp(P; qP; L) ∈ ℙ
|
BY GenUnivCD
|\
| 1. P: Term → ℙ
| 2. qP: Term
| 3. L: Term → ℙ
| ⊢ ReflectsProp(P; qP; L) ∈ ℙ
| |
1 BY Unfold ‘ReflectsProp‘ 0
THEN Unfold ‘guard‘ 0 THEN Auto
|\
| 1. P: Term → ℙ
| 2. qP: Term
| ⊢ Term → ℙ ∈ 𝕌'
| |
1 BY Auto
|\
| 1. P: Term → ℙ
| ⊢ Term ∈ 𝕌
| |
1 BY Auto
 \
  ⊢ Term → ℙ ∈ 𝕌'
  |
  BY Auto

<<< REPSTRUTH_WF >>>
⊢ ∀Tr:Term → ℙ. ∀tr:Term. ∀L:Term → ℙ.  RepsTruth(L; Tr; tr) ∈ ℙ
|
BY GenUnivCD
|\
| 1. Tr: Term → ℙ
| 2. tr: Term
| 3. L: Term → ℙ
| ⊢ RepsTruth(L; Tr; tr) ∈ ℙ
| |
1 BY Unfold ‘RepsTruth‘ 0 THEN Auto
|\
| 1. Tr: Term → ℙ
| 2. tr: Term
| ⊢ Term → ℙ ∈ 𝕌'
| |
1 BY Auto
|\
| 1. Tr: Term → ℙ
| ⊢ Term ∈ 𝕌
| |
1 BY Auto
```

```
 \
  ⊢ Term ⟶ ℙ ∈ 𝕌'
  |
  BY Auto

<<< PROP_AND_IFF >>>
⊢ ∀A,B,C:ℙ.  B ⇒ (A ⟺ B ∧ C) ⇒ {A ⟺ C}
|
BY Unfold ‘guard‘ 0 THEN ProvePropWith Auto

<<< PROP_IFF_TRANS >>>
⊢ ∀A,B,C:ℙ.  (A ⟺ B) ⇒ (B ⟺ C) ⇒ {A ⟺ C}
|
BY Unfold ‘guard‘ 0 THEN ProvePropWith Auto

<<< PROP_IFF_CONTRA >>>
⊢ ∀P:ℙ. (P ⟺ ¬P) ⇒ False
|
BY ProvePropWith Auto THEN Auto

<<< TARSKI >>>
⊢ ¬(∃Tr:Term ⟶ ℙ. ∃tr:Term. ∃L:Term ⟶ ℙ. RepsTruth(L; Tr; tr))
|
BY D 0 THENA Auto
|
1. ∃Tr:Term ⟶ ℙ. ∃tr:Term. ∃L:Term ⟶ ℙ. RepsTruth(L; Tr; tr)
⊢ False
|
BY Unfold ‘RepsTruth‘ 1 THEN ExRepD
|
1. Tr: Term ⟶ ℙ
2. tr: Term
3. L: Term ⟶ ℙ
4. ∀S:Term. (∃t:Term. S ↓= t) ⇒ L subx(tr; S)
5. RespectsNot(Tr; L)
6. ReflectsProp(Tr; tr; Tr)
|
BY Let ⌜S = s((¬tr)) ∈ Term⌝ THENA Auto
|
7. S: Term
8. S = s((¬tr)) ∈ Term
|
BY Assert ⌜S ↓= ¬subx(tr; S)⌝
|\
| ⊢ S ↓= ¬subx(tr; S)
| |
1 BY HypSubst (-1) 0 THENA Auto
| |
| ⊢ s((¬tr)) ↓= ¬subx(tr; s((¬tr)))
| |
1 BY RWHRevL ‘qnot_subx‘ 0 THEN Auto
| |
| ⊢ s((¬tr)) ↓= subx((¬tr); s((¬tr)))
| |
1 BY BLemma ‘s_reps‘ THEN Auto
 \
  9. S ↓= ¬subx(tr; S)
  |
```

```
  BY InstHyp [⌜S⌝] 4 THENW Auto
THENA (With ⌜¬subx(tr; S)⌝ (D 0) THEN Auto)
THEN Thin 4
  |
  4. RespectsNot(Tr; L)
  5. ReflectsProp(Tr; tr; Tr)
  6. S: Term
  7. S = s((¬tr)) ∈ Term
  8. S ↓= ¬subx(tr; S)
  9. L subx(tr; S)
  |
  BY Unfold ‘ReflectsProp‘ 5
THEN InstHyp [⌜¬subx(tr; S)⌝;⌜S⌝] 5
THENA Auto THEN Thin 5
  |
  5. S: Term
  6. S = s((¬tr)) ∈ Term
  7. S ↓= ¬subx(tr; S)
  8. L subx(tr; S)
  9. Tr subx(tr; S) ⟺ Tr ¬subx(tr; S)
  |
  BY Unfold ‘RespectsNot‘ 4
THEN InstHyp [⌜subx(tr; S)⌝] 4
THENA Auto THEN Thin 4
  |
  4. S: Term
  5. S = s((¬tr)) ∈ Term
  6. S ↓= ¬subx(tr; S)
  7. L subx(tr; S)
  8. Tr subx(tr; S) ⟺ Tr ¬subx(tr; S)
  9. Tr ¬subx(tr; S) ⟺ L subx(tr; S) ∧ ¬(Tr subx(tr; S))
  |
  BY Assert ⌜Tr ¬subx(tr; S) ⟺ ¬(Tr subx(tr; S))⌝
THENA (Using [‘B’,⌜L subx(tr; S)⌝]
             (BLemma ‘prop_and_iff‘)
       THEN Auto)
THEN OnHyps [9;7] Thin
  |
  7. Tr subx(tr; S) ⟺ Tr ¬subx(tr; S)
  8. Tr ¬subx(tr; S) ⟺ ¬(Tr subx(tr; S))
  |
  BY FLemma ‘prop_iff_trans‘ [7;8] THENA Auto
THEN OnHyps [8;7] Thin
  |
  7. Tr subx(tr; S) ⟺ ¬(Tr subx(tr; S))
  |
  BY FLemma ‘prop_iff_contra‘ [7] THEN Auto
```