```
rel_mng_2_lemma  THM

⊢  ∀ds,da:Collection(dec()), de:sig(), rho:Decl
|  , st1:Collection(SimpleType), e1:{1of([[de]] rho)}
|  , s,s':{[[ds]] rho}, a:[[st1]] rho, tr:trace_env([[da]] rho)
|  , l:Term List.
|  (∀i:ℕ||l||. trace_consistent(rho;da;tr.proj;l[i])) ⇒
|  (∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
|   ||ls|| = ||l||
|   & (∀i:ℕ. i<||l|| ⇒ ls[i] ∈ term_types(ds;st1;de;l[i]))
|   ⇒
|   list_accum(x,t.x([[t]] e1 s s' a tr);f;l) ∈ Prop)
| by RepeatFor 10 (D 0 THENA Auto)
|
1. ds : Collection(dec())
2. da : Collection(dec())
3. de : sig()
4. rho : Decl
5. st1 : Collection(SimpleType)
6. e1 : {1of([[de]] rho)}
7. s : {[[ds]] rho}
8. s' : {[[ds]] rho}
9. a : [[st1]] rho
10. tr : trace_env([[da]] rho)
⊢  ∀l:Term List.
|  (∀i:ℕ||l||. trace_consistent(rho;da;tr.proj;l[i])) ⇒
|  (∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
|   ||ls|| = ||l||
|   & (∀i:ℕ. i<||l|| ⇒ ls[i] ∈ term_types(ds;st1;de;l[i]))
|   ⇒
|   list_accum(x,t.x([[t]] e1 s s' a tr);f;l) ∈ Prop)
| by ...
|
|\
| 1. ds : Collection(dec())
| 2. da : Collection(dec())
| 3. de : sig()
| 4. rho : Decl
| 5. st1 : Collection(SimpleType)
| 6. {1of([[de]] rho)}
| 7. {[[ds]] rho}
| 8. {[[ds]] rho}
| 9. [[st1]] rho
| 10. tr : trace_env([[da]] rho)
| 11. Term List
| 12. ∀i:ℕ0. trace_consistent(rho;da;tr.proj;nil[i])
| ⊢  ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
|     ||ls|| = 0 & (∀i:ℕ. i<0 ⇒ ls[i] ∈ term_types(ds;st1;de;nil[i]))
|     ⇒
|     f ∈ Prop
|    by InductionOnList
| THEN Reduce 0 THEN Auto'
|
|      THEN ExRepD THEN Auto'·
|
```

```
|\
| 1. ds : Collection(dec())
| 2. da : Collection(dec())
| 3. de : sig()
| 4. rho : Decl
| 5. st1 : Collection(SimpleType)
| 6. e1 : {1of([[de]] rho)}
| 7. s : {[[ds]] rho}
| 8. s' : {[[ds]] rho}
| 9. a : [[st1]] rho
| 10. tr : trace_env([[da]] rho)
| 11. Term List
| 12. u : Term
| 13. v : Term List
| 14. (∀i:ℕ||v||. trace_consistent(rho;da;tr.proj;v[i])) ⇒
| |   (∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| |    ||ls|| = ||v||
| |    & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| |    ⇒
| |    list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop)
| 15. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| ⊢  ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| |  ||ls|| = ||v||+1
| |  & (∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i]))
| |  ⇒
| |
| |  list_accum(x,t.x([[t]] e1 s s' a tr);f([[u]] e1 s s' a tr);v)
| |  ∈ Prop  by UnivCD THENA (Auto' THEN Reduce 0 THEN Auto)
| |
| 1. ds : Collection(dec())
| 2. da : Collection(dec())
| 3. de : sig()
| 4. rho : Decl
| 5. st1 : Collection(SimpleType)
| 6. e1 : {1of([[de]] rho)}
| 7. s : {[[ds]] rho}
| 8. s' : {[[ds]] rho}
| 9. a : [[st1]] rho
| 10. tr : trace_env([[da]] rho)
| 11. Term List
| 12. u : Term
| 13. v : Term List
| 14. (∀i:ℕ||v||. trace_consistent(rho;da;tr.proj;v[i])) ⇒
| |   (∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| |    ||ls|| = ||v||
| |    & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| |    ⇒
| |    list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop)
| 15. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| 16. ls : SimpleType List
| 17. f : reduce(λs,m. [[s]] rho→m;Prop;ls)
| 18. ||ls|| = ||v||+1
| |   & (∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i]))
| ⊢  list_accum(x,t.x([[t]] e1 s s' a tr);f([[u]] e1 s s' a tr);v)
| |  ∈ Prop  by D (-1) THEN D (-6)
```

```
| .
| |
| |\
| | .....antecedent .....
| | 1. ds : Collection(dec())
| | 2. da : Collection(dec())
| | 3. de : sig()
| | 4. rho : Decl
| | 5. st1 : Collection(SimpleType)
| | 6. {1of([[de]] rho)}
| | 7. {[[ds]] rho}
| | 8. {[[ds]] rho}
| | 9. [[st1]] rho
| | 10. tr : trace_env([[da]] rho)
| | 11. Term List
| | 12. u : Term
| | 13. v : Term List
| | 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| | 15. ls : SimpleType List
| | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| | 17. ||ls|| = ||v||+1
| | 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| | ⊢  ∀i:ℕ||v||. trace_consistent(rho;da;tr.proj;v[i])
| | | by Auto THEN InstHyp [i+1] (-6) THENA Auto'
| | |THEN Subst'
| | |    [u / v][(i+1)] = v[i] (-1) THEN Auto'
| | |
| | 1. ds : Collection(dec())
| | 2. da : Collection(dec())
| | 3. de : sig()
| | 4. rho : Decl
| | 5. st1 : Collection(SimpleType)
| | 6. {1of([[de]] rho)}
| | 7. {[[ds]] rho}
| | 8. {[[ds]] rho}
| | 9. [[st1]] rho
| | 10. tr : trace_env([[da]] rho)
| | 11. Term List
| | 12. u : Term
| | 13. v : Term List
| | 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| | 15. ls : SimpleType List
| | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| | 17. ||ls|| = ||v||+1
| | 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| | 19. i : ℕ||v||
| | 20. trace_consistent(rho;da;tr.proj;[u / v][(i+1)])
| | 21. ∀x,y:Term. x = y ⇒ (x ∼ y)
| | ⊢  [u / v][(i+1)] = v[i]  by RWO "select_cons_tl" 0 THEN Auto'
| |
| .\
| . 1. ds : Collection(dec())
| . 2. da : Collection(dec())
| . 3. de : sig()
| . 4. rho : Decl
```

```
| . 5. st1 : Collection(SimpleType)
| . 6. e1 : {1of([[de]] rho)}
| . 7. s : {[[ds]] rho}
| . 8. s' : {[[ds]] rho}
| . 9. a : [[st1]] rho
| . 10. tr : trace_env([[da]] rho)
| . 11. Term List
| . 12. u : Term
| . 13. v : Term List
| . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . 15. ls : SimpleType List
| . 16. f : reduce(λs,m. [[s]] rho→m;Prop;ls)
| . 17. ||ls|| = ||v||+1
| . 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . |    ||ls|| = ||v||
| . |    & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . |    ⇒
| . |    list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . ⊢  list_accum(x,t.x([[t]] e1 s s' a tr);f([[u]] e1 s s' a tr);v)
| . |  ∈ Prop
| . | by AllHyps (\i.InstHyp [tl(ls)] i THENA Complete Auto)·
| . |
| . 1. ds : Collection(dec())
| . 2. da : Collection(dec())
| . 3. de : sig()
| . 4. rho : Decl
| . 5. st1 : Collection(SimpleType)
| . 6. e1 : {1of([[de]] rho)}
| . 7. s : {[[ds]] rho}
| . 8. s' : {[[ds]] rho}
| . 9. a : [[st1]] rho
| . 10. tr : trace_env([[da]] rho)
| . 11. Term List
| . 12. u : Term
| . 13. v : Term List
| . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . 15. ls : SimpleType List
| . 16. f : reduce(λs,m. [[s]] rho→m;Prop;ls)
| . 17. ||ls|| = ||v||+1
| . 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . |    ||ls|| = ||v||
| . |    & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . |    ⇒
| . |    list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . |    ||tl(ls)|| = ||v||
| . |    & (∀i:ℕ. i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . |    ⇒
| . |    list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . ⊢  list_accum(x,t.x([[t]] e1 s s' a tr);f([[u]] e1 s s' a tr);v)
| . |  ∈ Prop  by BackThruSomeHyp THEN Auto
| . |
| . |\
```

```
| . | .....wf .....
| . | 1. ds : Collection(dec())
| . | 2. da : Collection(dec())
| . | 3. de : sig()
| . | 4. rho : Decl
| . | 5. st1 : Collection(SimpleType)
| . | 6. e1 : {1of([[de]] rho)}
| . | 7. s : {[[ds]] rho}
| . | 8. s' : {[[ds]] rho}
| . | 9. a : [[st1]] rho
| . | 10. tr : trace_env([[da]] rho)
| . | 11. Term List
| . | 12. u : Term
| . | 13. v : Term List
| . | 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . | 15. ls : SimpleType List
| . | 16. f : reduce(λs,m. [[s]] rho→m;Prop;ls)
| . | 17. ||ls|| = ||v||+1
| . | 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . | 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . | |   ||ls|| = ||v||
| . | |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . | |   ⇒
| . | |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . | |   ||tl(ls)|| = ||v||
| . | |   & (∀i:ℕ. i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . | |   ⇒
| . | |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | ⊢  f([[u]] e1 s s' a tr)
| . | |  ∈ reduce(λs,m. [[s]] rho→m;Prop;tl(ls))
| . | | by Subst ls ~ [hd(ls) / tl(ls)] (-5)
| . | |
| . | |\
| . | | .....equality .....
| . | | 1. ds : Collection(dec())
| . | | 2. da : Collection(dec())
| . | | 3. de : sig()
| . | | 4. rho : Decl
| . | | 5. st1 : Collection(SimpleType)
| . | | 6. e1 : {1of([[de]] rho)}
| . | | 7. s : {[[ds]] rho}
| . | | 8. s' : {[[ds]] rho}
| . | | 9. a : [[st1]] rho
| . | | 10. tr : trace_env([[da]] rho)
| . | | 11. Term List
| . | | 12. u : Term
| . | | 13. v : Term List
| . | | 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . | | 15. ls : SimpleType List
| . | | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . | | 17. ||ls|| = ||v||+1
| . | | 18. ∀i:ℕ.
| . | |     i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . | | 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
```

```
| . | |     ||ls|| = ||v||
| . | |     & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . | |     ⇒
| . | |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | | 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . | |     ||tl(ls)|| = ||v||
| . | |     & (∀i:ℕ.
| . | |        i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . | |     ⇒
| . | |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | | ⊢  ls ~ [hd(ls) / tl(ls)]
| . | |   by BackThruLemma `list_decomp` THEN Auto'
| . | |
| . | .\
| . | . 1. ds : Collection(dec())
| . | . 2. da : Collection(dec())
| . | . 3. de : sig()
| . | . 4. rho : Decl
| . | . 5. st1 : Collection(SimpleType)
| . | . 6. e1 : {1of([[de]] rho)}
| . | . 7. s : {[[ds]] rho}
| . | . 8. s' : {[[ds]] rho}
| . | . 9. a : [[st1]] rho
| . | . 10. tr : trace_env([[da]] rho)
| . | . 11. Term List
| . | . 12. u : Term
| . | . 13. v : Term List
| . | . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . | . 15. ls : SimpleType List
| . | . 16. f : reduce(λs,m. [[s]] rho→m;Prop;[hd(ls) / tl(ls)])
| . | . 17. ||ls|| = ||v||+1
| . | . 18. ∀i:ℕ.
| . | . |   i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . | . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . | . |   ||ls|| = ||v||
| . | . |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . | . |   ⇒
| . | . |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . | . |   ||tl(ls)|| = ||v||
| . | . |   & (∀i:ℕ.
| . | . |      i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . | . |   ⇒
| . | . |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | . ⊢  f([[u]] e1 s s' a tr)
| . | . |  ∈ reduce(λs,m. [[s]] rho→m;Prop;tl(ls))
| . | . | by RecUnfold `reduce` (-5) THEN Reduce (-5) THEN Auto
| . | . |
| . | . .....subterm .....
| . | . 1. ds : Collection(dec())
| . | . 2. da : Collection(dec())
| . | . 3. de : sig()
| . | . 4. rho : Decl
| . | . 5. st1 : Collection(SimpleType)
| . | . 6. e1 : {1of([[de]] rho)}
```

```
| . | . 7. s : {[[ds]] rho}
| . | . 8. s' : {[[ds]] rho}
| . | . 9. a : [[st1]] rho
| . | . 10. tr : trace_env([[da]] rho)
| . | . 11. Term List
| . | . 12. u : Term
| . | . 13. v : Term List
| . | . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . | . 15. ls : SimpleType List
| . | . 16. [[hd(ls)]] rho→reduce(λs,m. [[s]] rho→m;Prop;tl(ls))
| . | . 17. ||ls|| = ||v||+1
| . | . 18. ∀i:ℕ.
| . | . |   i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . | . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . | . |   ||ls|| = ||v||
| . | . |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . | . |   ⇒
| . | . |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . | . |   ||tl(ls)|| = ||v||
| . | . |   & (∀i:ℕ.
| . | . |      i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . | . |   ⇒
| . | . |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | . ⊢  [[u]] e1 s s' a tr ∈ [[hd(ls)]] rho
| . | . | by BackThruLemma `term_typing2` THENA Auto'
| . | .
| . | . |     THEN AllHyps (\h. InstHyp [0] h THENA (Complete Auto')
| . | . |
| . | . |            THEN Reduce (-1) THEN Trivial)
| . | .
| . | . |    THEN Try (InstHyp [0] (-3) THENA Auto'
| . | .
| . | . |              THEN Unfold `select` (-1) THEN  Reduce (-1)
| . | .
| . | . |              THEN Try Trivial)·
| . | . |
| . | . 1. ds : Collection(dec())
| . | . 2. da : Collection(dec())
| . | . 3. de : sig()
| . | . 4. rho : Decl
| . | . 5. st1 : Collection(SimpleType)
| . | . 6. e1 : {1of([[de]] rho)}
| . | . 7. s : {[[ds]] rho}
| . | . 8. s' : {[[ds]] rho}
| . | . 9. a : [[st1]] rho
| . | . 10. tr : trace_env([[da]] rho)
| . | . 11. Term List
| . | . 12. u : Term
| . | . 13. v : Term List
| . | . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . | . 15. ls : SimpleType List
| . | . 16. [[hd(ls)]] rho→reduce(λs,m. [[s]] rho→m;Prop;tl(ls))
| . | . 17. ||ls|| = ||v||+1
| . | . 18. ∀i:ℕ.
```

```
| . | .     i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . | . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . | .     ||ls|| = ||v||
| . | .     & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . | .     ⇒
| . | .     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . | .     ||tl(ls)|| = ||v||
| . | .     & (∀i:ℕ.
| . | .        i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . | .     ⇒
| . | .     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | . 21. hd(ls) ∈ term_types(ds;st1;de;u)
| . | . ⊢  ls ∈ SimpleType List+
| . | .   by ...
| . |
| . |\
| . | 1. ds : Collection(dec())
| . | 2. da : Collection(dec())
| . | 3. de : sig()
| . | 4. rho : Decl
| . | 5. st1 : Collection(SimpleType)
| . | 6. e1 : {1of([[de]] rho)}
| . | 7. s : {[[ds]] rho}
| . | 8. s' : {[[ds]] rho}
| . | 9. a : [[st1]] rho
| . | 10. tr : trace_env([[da]] rho)
| . | 11. Term List
| . | 12. u : Term
| . | 13. v : Term List
| . | 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . | 15. ls : SimpleType List
| . | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . | 17. ||ls|| = ||v||+1
| . | 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . | 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . |     ||ls|| = ||v||
| . |     & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . |     ⇒
| . |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . |     ||tl(ls)|| = ||v||
| . |     & (∀i:ℕ. i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . |     ⇒
| . |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . | ⊢  ||tl(ls)|| = ||v||
| . |   by InstLemma `length_tl` [SimpleType;ls] THEN Auto'
| . |
| . .\
| . . 1. ds : Collection(dec())
| . . 2. da : Collection(dec())
| . . 3. de : sig()
| . . 4. rho : Decl
| . . 5. st1 : Collection(SimpleType)
| . . 6. e1 : {1of([[de]] rho)}
```

```
| . . 7. s : {[[ds]] rho}
| . . 8. s’ : {[[ds]] rho}
| . . 9. a : [[st1]] rho
| . . 10. tr : trace_env([[da]] rho)
| . . 11. Term List
| . . 12. u : Term
| . . 13. v : Term List
| . . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . . 15. ls : SimpleType List
| . . 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . 17. ||ls|| = ||v||+1
| . . 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . . |   ||ls|| = ||v||
| . . |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . |   ⇒
| . . |   list_accum(x,t.x([[t]] e1 s s’ a tr);f;v) ∈ Prop
| . . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . |   ||tl(ls)|| = ||v||
| . . |   & (∀i:ℕ. i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . |   ⇒
| . . |   list_accum(x,t.x([[t]] e1 s s’ a tr);f;v) ∈ Prop
| . . 21. i : ℕ
| . . 22. i<||v||
| . . ⊢  tl(ls)[i] ∈ term_types(ds;st1;de;v[i])
| . . | by InstHyp [i+1] (-5) THENA Auto
| . . |
| . . 1. ds : Collection(dec())
| . . 2. da : Collection(dec())
| . . 3. de : sig()
| . . 4. rho : Decl
| . . 5. st1 : Collection(SimpleType)
| . . 6. e1 : {1of([[de]] rho)}
| . . 7. s : {[[ds]] rho}
| . . 8. s’ : {[[ds]] rho}
| . . 9. a : [[st1]] rho
| . . 10. tr : trace_env([[da]] rho)
| . . 11. Term List
| . . 12. u : Term
| . . 13. v : Term List
| . . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . . 15. ls : SimpleType List
| . . 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . 17. ||ls|| = ||v||+1
| . . 18. ∀i:ℕ. i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . . |   ||ls|| = ||v||
| . . |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . |   ⇒
| . . |   list_accum(x,t.x([[t]] e1 s s’ a tr);f;v) ∈ Prop
| . . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . |   ||tl(ls)|| = ||v||
| . . |   & (∀i:ℕ. i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . |   ⇒
| . . |   list_accum(x,t.x([[t]] e1 s s’ a tr);f;v) ∈ Prop
```

```
| . . 21. i : ℕ
| . . 22. i<||v||
| . . 23. ls[(i+1)] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . ⊢  tl(ls)[i] ∈ term_types(ds;st1;de;v[i])
| . . | by Subst ls[(i+1)] ~ tl(ls)[i] (-1)
| . . |
| . . |\
| . . | .....equality .....
| . . | 1. ds : Collection(dec())
| . . | 2. da : Collection(dec())
| . . | 3. de : sig()
| . . | 4. rho : Decl
| . . | 5. st1 : Collection(SimpleType)
| . . | 6. e1 : {1of([[de]] rho)}
| . . | 7. s : {[[ds]] rho}
| . . | 8. s' : {[[ds]] rho}
| . . | 9. a : [[st1]] rho
| . . | 10. tr : trace_env([[da]] rho)
| . . | 11. Term List
| . . | 12. u : Term
| . . | 13. v : Term List
| . . | 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . . | 15. ls : SimpleType List
| . . | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . | 17. ||ls|| = ||v||+1
| . . | 18. ∀i:ℕ.
| . . | |   i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . | 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . . | |   ||ls|| = ||v||
| . . | |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . | |   ⇒
| . . | |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . | 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . | |   ||tl(ls)|| = ||v||
| . . | |   & (∀i:ℕ.
| . . | |      i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . | |   ⇒
| . . | |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . | 21. i : ℕ
| . . | 22. i<||v||
| . . | 23. ls[(i+1)] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . | ⊢  ls[(i+1)] ~ tl(ls)[i]
| . . | | by Unfold `select` 0 THEN EqCD THEN
| . . |
| . . | |    RW (AddrC [1] (RecUnfoldC `nth_tl`)) 0
| . . |
| . . | |    THEN SplitOnConclITE THENA Auto
| . . | |
| . . | |\
| . . | | .....truecase .....
| . . | | 1. ds : Collection(dec())
| . . | | 2. da : Collection(dec())
| . . | | 3. de : sig()
| . . | | 4. rho : Decl
| . . | | 5. st1 : Collection(SimpleType)
```

```
| . . | | 6. e1 : {1of([[de]] rho)}
| . . | | 7. s : {[[ds]] rho}
| . . | | 8. s' : {[[ds]] rho}
| . . | | 9. a : [[st1]] rho
| . . | | 10. tr : trace_env([[da]] rho)
| . . | | 11. Term List
| . . | | 12. u : Term
| . . | | 13. v : Term List
| . . | | 14. ∀i:ℕ(||v||+1).
| . . | |     trace_consistent(rho;da;tr.proj;[u / v][i])
| . . | | 15. ls : SimpleType List
| . . | | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . | | 17. ||ls|| = ||v||+1
| . . | | 18. ∀i:ℕ.
| . . | |     i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . | | 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→
| . . | |                              m;Prop;ls).
| . . | |     ||ls|| = ||v||
| . . | |     & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . | |     ⇒
| . . | |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . | | 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . | |     ||tl(ls)|| = ||v||
| . . | |     & (∀i:ℕ.
| . . | |        i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . | |     ⇒
| . . | |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . | | 21. i : ℕ
| . . | | 22. i<||v||
| . . | | 23. ls[(i+1)] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . | | 24. i+1≤0
| . . | | ⊢  ls ~ nth_tl(i;tl(ls))  by Auto
| . . | |
| . . | .\
| . . | . .....falsecase .....
| . . | . 1. ds : Collection(dec())
| . . | . 2. da : Collection(dec())
| . . | . 3. de : sig()
| . . | . 4. rho : Decl
| . . | . 5. st1 : Collection(SimpleType)
| . . | . 6. e1 : {1of([[de]] rho)}
| . . | . 7. s : {[[ds]] rho}
| . . | . 8. s' : {[[ds]] rho}
| . . | . 9. a : [[st1]] rho
| . . | . 10. tr : trace_env([[da]] rho)
| . . | . 11. Term List
| . . | . 12. u : Term
| . . | . 13. v : Term List
| . . | . 14. ∀i:ℕ(||v||+1).
| . . | .     trace_consistent(rho;da;tr.proj;[u / v][i])
| . . | . 15. ls : SimpleType List
| . . | . 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . | . 17. ||ls|| = ||v||+1
| . . | . 18. ∀i:ℕ.
| . . | .     i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
```

```
| . . | . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→
| . . | .                           m;Prop;ls).
| . . | .     ||ls|| = ||v||
| . . | .     & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . | .     ⇒
| . . | .     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . | . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . | .     ||tl(ls)|| = ||v||
| . . | .     & (∀i:ℕ.
| . . | .        i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . | .     ⇒
| . . | .     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . | . 21. i : ℕ
| . . | . 22. i<||v||
| . . | . 23. ls[(i+1)] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . | . 24. 0<i+1
| . . | . ⊢  nth_tl(i+1-1;tl(ls)) ~ nth_tl(i;tl(ls))
| . . | .   by EqCD THEN Auto'
| . . |
| . . .\
| . . . 1. ds : Collection(dec())
| . . . 2. da : Collection(dec())
| . . . 3. de : sig()
| . . . 4. rho : Decl
| . . . 5. st1 : Collection(SimpleType)
| . . . 6. e1 : {1of([[de]] rho)}
| . . . 7. s : {[[ds]] rho}
| . . . 8. s' : {[[ds]] rho}
| . . . 9. a : [[st1]] rho
| . . . 10. tr : trace_env([[da]] rho)
| . . . 11. Term List
| . . . 12. u : Term
| . . . 13. v : Term List
| . . . 14. ∀i:ℕ(||v||+1). trace_consistent(rho;da;tr.proj;[u / v][i])
| . . . 15. ls : SimpleType List
| . . . 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . . 17. ||ls|| = ||v||+1
| . . . 18. ∀i:ℕ.
| . . . |   i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
| . . . |   ||ls|| = ||v||
| . . . |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . . |   ⇒
| . . . |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . . |   ||tl(ls)|| = ||v||
| . . . |   & (∀i:ℕ.
| . . . |      i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . . |   ⇒
| . . . |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . 21. i : ℕ
| . . . 22. i<||v||
| . . . 23. tl(ls)[i] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . . ⊢  tl(ls)[i] ∈ term_types(ds;st1;de;v[i])
| . . . | by Subst [u / v][(i+1)] ~ v[i] (-1)
```

```
| . . . |
| . . . |\
| . . . | .....equality .....
| . . . | 1. ds : Collection(dec())
| . . . | 2. da : Collection(dec())
| . . . | 3. de : sig()
| . . . | 4. rho : Decl
| . . . | 5. st1 : Collection(SimpleType)
| . . . | 6. e1 : {1of([[de]] rho)}
| . . . | 7. s : {[[ds]] rho}
| . . . | 8. s' : {[[ds]] rho}
| . . . | 9. a : [[st1]] rho
| . . . | 10. tr : trace_env([[da]] rho)
| . . . | 11. Term List
| . . . | 12. u : Term
| . . . | 13. v : Term List
| . . . | 14. ∀i:ℕ(||v||+1).
| . . . | |   trace_consistent(rho;da;tr.proj;[u / v][i])
| . . . | 15. ls : SimpleType List
| . . . | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . . | 17. ||ls|| = ||v||+1
| . . . | 18. ∀i:ℕ.
| . . . | |   i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . . | 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→
| . . . | |                             m;Prop;ls).
| . . . | |   ||ls|| = ||v||
| . . . | |   & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . . | |   ⇒
| . . . | |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . | 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . . | |   ||tl(ls)|| = ||v||
| . . . | |   & (∀i:ℕ.
| . . . | |      i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . . | |   ⇒
| . . . | |   list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . | 21. i : ℕ
| . . . | 22. i<||v||
| . . . | 23. tl(ls)[i] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . . | ⊢  [u / v][(i+1)] ~ v[i]
| . . . | | by Unfold `select` 0 THEN EqCD
| . . . |
| . . . | |    THEN RW (AddrC [1] (RecUnfoldC `nth_tl`)) 0

| . . . |
| . . . | |    THEN SplitOnConclITE THENA Auto
| . . . | |
| . . . | |\
| . . . | | .....truecase .....
| . . . | | 1. ds : Collection(dec())
| . . . | | 2. da : Collection(dec())
| . . . | | 3. de : sig()
| . . . | | 4. rho : Decl
| . . . | | 5. st1 : Collection(SimpleType)
| . . . | | 6. e1 : {1of([[de]] rho)}
| . . . | | 7. s : {[[ds]] rho}
| . . . | | 8. s' : {[[ds]] rho}
```

```
| . . . | | 9. a : [[st1]] rho
| . . . | | 10. tr : trace_env([[da]] rho)
| . . . | | 11. Term List
| . . . | | 12. u : Term
| . . . | | 13. v : Term List
| . . . | | 14. ∀i:ℕ(||v||+1).
| . . . | |     trace_consistent(rho;da;tr.proj;[u / v][i])
| . . . | | 15. ls : SimpleType List
| . . . | | 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . . | | 17. ||ls|| = ||v||+1
| . . . | | 18. ∀i:ℕ.
| . . . | |     i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . . | | 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→
| . . . | |                              m;Prop;ls).
| . . . | |     ||ls|| = ||v||
| . . . | |     & (∀i:ℕ.
| . . . | |        i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . . | |     ⇒
| . . . | |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . | | 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . . | |     ||tl(ls)|| = ||v||
| . . . | |     & (∀i:ℕ.
| . . . | |        i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . . | |     ⇒
| . . . | |     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . | | 21. i : ℕ
| . . . | | 22. i<||v||
| . . . | | 23. tl(ls)[i] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . . | | 24. i+1≤0
| . . . | | ⊢  [u / v] ∼ nth_tl(i;v)  by Auto
| . . . | |
| . . . | .\
| . . . | . .....falsecase .....
| . . . | . 1. ds : Collection(dec())
| . . . | . 2. da : Collection(dec())
| . . . | . 3. de : sig()
| . . . | . 4. rho : Decl
| . . . | . 5. st1 : Collection(SimpleType)
| . . . | . 6. e1 : {1of([[de]] rho)}
| . . . | . 7. s : {[[ds]] rho}
| . . . | . 8. s' : {[[ds]] rho}
| . . . | . 9. a : [[st1]] rho
| . . . | . 10. tr : trace_env([[da]] rho)
| . . . | . 11. Term List
| . . . | . 12. u : Term
| . . . | . 13. v : Term List
| . . . | . 14. ∀i:ℕ(||v||+1).
| . . . | .     trace_consistent(rho;da;tr.proj;[u / v][i])
| . . . | . 15. ls : SimpleType List
| . . . | . 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . . | . 17. ||ls|| = ||v||+1
| . . . | . 18. ∀i:ℕ.
| . . . | .     i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . . | . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→
| . . . | .                              m;Prop;ls).
```

```
| . . . . | .     ||ls|| = ||v||
| . . . . | .     & (∀i:ℕ.
| . . . . | .        i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . . . | .     ⇒
| . . . . | .     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . . | . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . . . | .     ||tl(ls)|| = ||v||
| . . . . | .     & (∀i:ℕ.
| . . . . | .        i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . . . | .     ⇒
| . . . . | .     list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . . | . 21. i : ℕ
| . . . . | . 22. i<||v||
| . . . . | . 23. tl(ls)[i] ∈ term_types(ds;st1;de;[u / v][(i+1)])
| . . . . | . 24. 0<i+1
| . . . . | . ⊢  nth_tl(i+1-1;tl([u / v])) ~ nth_tl(i;v)
| . . . . | .   by Reduce 0 THEN EqCD THEN Auto
| . . . . |
| . . . . .\
| . . . . . 1. ds : Collection(dec())
| . . . . . 2. da : Collection(dec())
| . . . . . 3. de : sig()
| . . . . . 4. rho : Decl
| . . . . . 5. st1 : Collection(SimpleType)
| . . . . . 6. e1 : {1of([[de]] rho)}
| . . . . . 7. s : {[[ds]] rho}
| . . . . . 8. s' : {[[ds]] rho}
| . . . . . 9. a : [[st1]] rho
| . . . . . 10. tr : trace_env([[da]] rho)
| . . . . . 11. Term List
| . . . . . 12. u : Term
| . . . . . 13. v : Term List
| . . . . . 14. ∀i:ℕ(||v||+1).
| . . . .       trace_consistent(rho;da;tr.proj;[u / v][i])
| . . . . . 15. ls : SimpleType List
| . . . . . 16. reduce(λs,m. [[s]] rho→m;Prop;ls)
| . . . . . 17. ||ls|| = ||v||+1
| . . . . . 18. ∀i:ℕ.
| . . . .       i<||v||+1 ⇒ ls[i] ∈ term_types(ds;st1;de;[u / v][i])
| . . . . . 19. ∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→
| . . . .                                 m;Prop;ls).
| . . . .       ||ls|| = ||v||
| . . . .       & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
| . . . .       ⇒
| . . . .       list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . . . 20. ∀f:reduce(λs,m. [[s]] rho→m;Prop;tl(ls)).
| . . . .       ||tl(ls)|| = ||v||
| . . . .       & (∀i:ℕ.
| . . . .          i<||v|| ⇒ tl(ls)[i] ∈ term_types(ds;st1;de;v[i]))
| . . . .       ⇒
| . . . .       list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop
| . . . . . 21. i : ℕ
| . . . . . 22. i<||v||
| . . . . . 23. tl(ls)[i] ∈ term_types(ds;st1;de;v[i])
| . . . . . ⊢  tl(ls)[i] ∈ term_types(ds;st1;de;v[i])  by Trivial
```

```
|
.\
. 1. ds : Collection(dec())
. 2. da : Collection(dec())
. 3. de : sig()
. 4. rho : Decl
. 5. st1 : Collection(SimpleType)
. 6. e1 : {1of([[de]] rho)}
. 7. s : {[[ds]] rho}
. 8. s' : {[[ds]] rho}
. 9. a : [[st1]] rho
. 10. tr : trace_env([[da]] rho)
. 11. Term List
. 12. u : Term
. 13. v : Term List
. 14. (∀i:ℕ||v||. trace_consistent(rho;da;tr.proj;v[i])) ⇒
.     (∀ls:SimpleType List, f:reduce(λs,m. [[s]] rho→m;Prop;ls).
.      ||ls|| = ||v||
.      & (∀i:ℕ. i<||v|| ⇒ ls[i] ∈ term_types(ds;st1;de;v[i]))
.      ⇒
.      list_accum(x,t.x([[t]] e1 s s' a tr);f;v) ∈ Prop)
. 15. i : ℕ(||v||+1)
. ⊢  i<||[u / v]||  by Auto
END
```