```
select_zip  THM

⊢  ∀T1,T2:Type, as:T1 List, bs:T2 List, i:ℕ.
|  i<||zip(as;bs)|| ⇒ zip(as;bs)[i] = <as[i],bs[i]>
| by D 0 THENM D 0 THENM D 0 THENA Auto
THEN ListInd (-1)

|    THEN D 0 THENA Auto
THEN ListInd (-1)
THEN RecUnfold ‘zip‘ 0
|
THEN Reduce 0 THEN Try (Complete Auto)
|
1. T1 : Type
2. T2 : Type
3. T1 List
4. u : T1
5. v : T1 List
6. ∀bs:T2 List, i:ℕ. i<||zip(v;bs)|| ⇒ zip(v;bs)[i] = <v[i],bs[i]>
7. T2 List
8. u1 : T2
9. v1 : T2 List
10. ∀i:ℕ.
|   i<||zip([u / v];v1)|| ⇒ zip([u / v];v1)[i] = <[u / v][i],v1[i]>
⊢  ∀i:ℕ.
|  i<||zip(v;v1)||+1
|  ⇒
|  [<u,u1> / zip(v;v1)][i] = <[u / v][i],[u1 / v1][i]>
| by UnivCD THENA Auto THEN
CaseNat 0 ‘i’
|
|\
| 1. T1 : Type
| 2. T2 : Type
| 3. T1 List
| 4. u : T1
| 5. v : T1 List
| 6. ∀bs:T2 List, i:ℕ. i<||zip(v;bs)|| ⇒ zip(v;bs)[i] = <v[i],bs[i]>
| 7. T2 List
| 8. u1 : T2
| 9. v1 : T2 List
| 10. ∀i:ℕ.
|     i<||zip([u / v];v1)|| ⇒ zip([u / v];v1)[i] = <[u / v][i],v1[i]>
| 11. i : ℕ
| 12. i<||zip(v;v1)||+1
| 13. i = 0
| ⊢  [<u,u1> / zip(v;v1)][0] = <[u / v][0],[u1 / v1][0]>
|   by Reduce 0 THEN Auto
|
.\
. 1. T1 : Type
. 2. T2 : Type
. 3. T1 List
. 4. u : T1
. 5. v : T1 List
```

```
. 6. ∀bs:T2 List, i:ℕ. i<||zip(v;bs)|| ⇒ zip(v;bs)[i] = <v[i],bs[i]>
. 7. T2 List
. 8. u1 : T2
. 9. v1 : T2 List
. 10. ∀i:ℕ.
.      i<||zip([u / v];v1)|| ⇒ zip([u / v];v1)[i] = <[u / v][i],v1[i]>
. 11. i : ℕ
. 12. i<||zip(v;v1)||+1
. 13. ¬i = 0
. ⊢  [<u,u1> / zip(v;v1)][i] = <[u / v][i],[u1 / v1][i]>
.   by InstLemma ‘zip_length‘ [T1;T2;v;v1]
.
.      THENM RWW "select_cons_tl" 0
. THENM BackThruSomeHyp
.
.      THEN Auto’
END
```