```
M_DASH_C_DASH_S_SPACE_induction  THM

⊢  ∀E:TaggedEventStruct, P,I:TraceProperty(E).
|  MCS(E)(P) ⇒
|  safetyR(E) preserves I ⇒
|  (I refines single-tag-decomposable(E)) ⇒ (I fuses P)
| by ...

|    ...
THEN RepeatFor 6 (D 0 THENA Auto)
THEN ExRepDHyps

|    THEN Assert ∀n:ℕ, tr:|E| List.
|    ||tr||≤n ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)

|    THEN Try (D 0 THENA Auto THEN Using [‘n’,||tr||
|    ] BackThruSomeHyp THEN Auto’)
THEN Try (InductionOnNat)
|
|\
| .....basecase .....
| 1. E : TaggedEventStruct
| 2. P : (|E| List)→Prop
| 3. I : (|E| List)→Prop
| 4. memorylessR(E) preserves P
| 5. (ternary) composableR(E) preserves P
| 6. safetyR(E) preserves P
| 7. safetyR(E) preserves I
| 8. ∀tr:|E| List.
| |  I(tr) ⇒
| |  ¬tr = nil ⇒
| |  (∃L_1,L_2:|E| List.
| |   tr = (L_1 @ L_2)
| |   & ¬L_2 = nil
| |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
| |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
| ⊢  ∀tr:|E| List. ||tr||≤0 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
| | by Auto THEN Subst tr = nil 0 THEN Auto
|
| |    THEN  Try (D 0 THEN Auto)
|
| |    THEN Try (BackThruLemma ‘length_zero‘ THEN Auto’)
| |
| 1. E : TaggedEventStruct
| 2. P : (|E| List)→Prop
| 3. I : (|E| List)→Prop
| 4. memorylessR(E) preserves P
| 5. (ternary) composableR(E) preserves P
| 6. safetyR(E) preserves P
| 7. safetyR(E) preserves I
| 8. ∀tr:|E| List.
|    I(tr) ⇒
|    ¬tr = nil ⇒
|    (∃L_1,L_2:|E| List.
|     tr = (L_1 @ L_2)
```

```
|     & ¬L_2 = nil
|     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
|     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
| 9. tr : |E| List
| 10. ||tr||≤0
| 11. ∀m:Label. P(<nil>_m)
| 12. I(nil)
| ⊢  P(nil)
|   by InstHyp [any] (-2) THENA Auto
|
|      THEN Unfold ‘tag_sublist‘ (-1) THEN Reduce (-1)
| THEN Auto
|
.\
. .....upcase .....
. 1. E : TaggedEventStruct
. 2. P : (|E| List)→Prop
. 3. I : (|E| List)→Prop
. 4. memorylessR(E) preserves P
. 5. (ternary) composableR(E) preserves P
. 6. safetyR(E) preserves P
. 7. safetyR(E) preserves I
. 8. ∀tr:|E| List.
. |  I(tr) ⇒
. |  ¬tr = nil ⇒
. |  (∃L_1,L_2:|E| List.
. |   tr = (L_1 @ L_2)
. |   & ¬L_2 = nil
. |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. 9. n : ℤ
. 10. 0<n
. 11. ∀tr:|E| List.
. |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. ⊢  ∀tr:|E| List. ||tr||≤n ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. | by Auto THEN Decide null(tr) THENA Auto
.
. |    THEN RW assert_pushdownC (-1) THENA Auto
.
. |    THEN Unfold ‘guard‘ 0
. THEN Try (HypSubst  (-1) 0 THEN Auto)
. |
. |\
. | 1. E : TaggedEventStruct
. | 2. P : (|E| List)→Prop
. | 3. I : (|E| List)→Prop
. | 4. memorylessR(E) preserves P
. | 5. (ternary) composableR(E) preserves P
. | 6. safetyR(E) preserves P
. | 7. safetyR(E) preserves I
. | 8. ∀tr:|E| List.
. |    I(tr) ⇒
. |    ¬tr = nil ⇒
. |    (∃L_1,L_2:|E| List.
. |     tr = (L_1 @ L_2)
```

```
. |     & ¬L_2 = nil
. |     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. |     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. | 9. n : ℤ
. | 10. 0<n
. | 11. ∀tr:|E| List.
. |     ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. | 12. tr : |E| List
. | 13. ||tr||≤n
. | 14. tr = nil
. | 15. ∀m:Label. P(<nil>_m)
. | 16. I(nil)
. | ⊢  P(nil)
. |   by InstHyp [any] (-2) THENA Auto
. |
. |      THEN Unfold `tag_sublist` (-1) THEN Reduce (-1)
. | THEN Auto
. |
. .\
. . 1. E : TaggedEventStruct
. . 2. P : (|E| List)→Prop
. . 3. I : (|E| List)→Prop
. . 4. memorylessR(E) preserves P
. . 5. (ternary) composableR(E) preserves P
. . 6. safetyR(E) preserves P
. . 7. safetyR(E) preserves I
. . 8. ∀tr:|E| List.
. . |  I(tr) ⇒
. . |  ¬tr = nil ⇒
. . |  (∃L_1,L_2:|E| List.
. . |   tr = (L_1 @ L_2)
. . |   & ¬L_2 = nil
. . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . 9. n : ℤ
. . 10. 0<n
. . 11. ∀tr:|E| List.
. . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . 12. tr : |E| List
. . 13. ||tr||≤n
. . 14. ¬tr = nil
. . ⊢  (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | by Auto THEN AllHyps (\h. InstHyp [tr] h THENA Complete Auto)
. .
. . |    THEN ExRepD THEN Auto
. .
. . |    ...
. . |                 THEN InstHyp [L_1 ;L_2;tr
. . |    ] h THEN Auto)
. . |
. . |\
. . | .....antecedent .....
. . | 1. E : TaggedEventStruct
. . | 2. P : (|E| List)→Prop
. . | 3. I : (|E| List)→Prop
```

```
. . | 4. memorylessR(E) preserves P
. . | 5. ∀x,y,z:|E| List.
. . | |  P(x) ⇒
. . | |  P(y) ⇒ (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | 6. safetyR(E) preserves P
. . | 7. safetyR(E) preserves I
. . | 8. ∀tr:|E| List.
. . | |  I(tr) ⇒
. . | |  ¬tr = nil ⇒
. . | |  (∃L_1,L_2:|E| List.
. . | |   tr = (L_1 @ L_2)
. . | |   & ¬L_2 = nil
. . | |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | 9. n : ℤ
. . | 10. 0<n
. . | 11. ∀tr:|E| List.
. . | |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | 12. tr : |E| List
. . | 13. ||tr||≤n
. . | 14. ¬tr = nil
. . | 15. ∀m:Label. P(<tr>_m)
. . | 16. I(tr)
. . | 17. L_1 : |E| List
. . | 18. L_2 : |E| List
. . | 19. tr = (L_1 @ L_2)
. . | 20. ¬L_2 = nil
. . | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | 22. m : Label
. . | 23. (∀x∈L_2.tag(E)(x) = m)
. . | ⊢  P(L_1)  by Assert ||L_1||≤n-1
. . | |
. . | |\
. . | | .....assertion .....
. . | | 1. E : TaggedEventStruct
. . | | 2. P : (|E| List)→Prop
. . | | 3. I : (|E| List)→Prop
. . | | 4. memorylessR(E) preserves P
. . | | 5. ∀x,y,z:|E| List.
. . | |    P(x) ⇒
. . | |    P(y) ⇒ (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | | 6. safetyR(E) preserves P
. . | | 7. safetyR(E) preserves I
. . | | 8. ∀tr:|E| List.
. . | |    I(tr) ⇒
. . | |    ¬tr = nil ⇒
. . | |    (∃L_1,L_2:|E| List.
. . | |     tr = (L_1 @ L_2)
. . | |     & ¬L_2 = nil
. . | |     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | |     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | | 9. n : ℤ
. . | | 10. 0<n
. . | | 11. ∀tr:|E| List.
. . | |     ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
```

```
. . | | 12. tr : |E| List
. . | | 13. ||tr||≤n
. . | | 14. ¬tr = nil
. . | | 15. ∀m:Label. P(<tr>_m)
. . | | 16. I(tr)
. . | | 17. L_1 : |E| List
. . | | 18. L_2 : |E| List
. . | | 19. tr = (L_1 @ L_2)
. . | | 20. ¬L_2 = nil
. . | | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | | 22. m : Label
. . | | 23. (∀x∈L_2.tag(E)(x) = m)
. . | | ⊢  ||L_1||≤n-1
. . | |   by InstLemma ‘length_append‘ [|E|;L_1;L_2] THENA Auto
. . | |
. . | |      THEN InstLemma ‘non_nil_length‘ [|E|;L_2] THENA Auto
. . | |
. . | |      THEN SubstFor L_1 @ L_2 (-2) THEN Auto’
. . | |
. . | .\
. . | . 1. E : TaggedEventStruct
. . | . 2. P : (|E| List)→Prop
. . | . 3. I : (|E| List)→Prop
. . | . 4. memorylessR(E) preserves P
. . | . 5. ∀x,y,z:|E| List.
. . | . |  P(x) ⇒
. . | . |  P(y) ⇒ (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | . 6. safetyR(E) preserves P
. . | . 7. safetyR(E) preserves I
. . | . 8. ∀tr:|E| List.
. . | . |  I(tr) ⇒
. . | . |  ¬tr = nil ⇒
. . | . |  (∃L_1,L_2:|E| List.
. . | . |   tr = (L_1 @ L_2)
. . | . |   & ¬L_2 = nil
. . | . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | . 9. n : ℤ
. . | . 10. 0<n
. . | . 11. ∀tr:|E| List.
. . | . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | . 12. tr : |E| List
. . | . 13. ||tr||≤n
. . | . 14. ¬tr = nil
. . | . 15. ∀m:Label. P(<tr>_m)
. . | . 16. I(tr)
. . | . 17. L_1 : |E| List
. . | . 18. L_2 : |E| List
. . | . 19. tr = (L_1 @ L_2)
. . | . 20. ¬L_2 = nil
. . | . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . 22. m : Label
. . | . 23. (∀x∈L_2.tag(E)(x) = m)
. . | . 24. ||L_1||≤n-1
. . | . ⊢  P(L_1)
```

```
. . | . | by AssertBY L_1 ≤ tr (Unfold ‘iseg‘ 0 THEN InstConcl [L_2
. . | . |    ] THEN Auto)
. . | . THEN AssertBY I(L_1)
. . | .
. . | . |    ...
. . | . |     THEN Using [‘x’,tr
. . | . |    ] (BackThruHyp h) THEN Auto))
. . | . |
. . | . 1. E : TaggedEventStruct
. . | . 2. P : (|E| List)→Prop
. . | . 3. I : (|E| List)→Prop
. . | . 4. memorylessR(E) preserves P
. . | . 5. ∀x,y,z:|E| List.
. . | . |  P(x) ⇒
. . | . |  P(y) ⇒ (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | . 6. safetyR(E) preserves P
. . | . 7. safetyR(E) preserves I
. . | . 8. ∀tr:|E| List.
. . | . |  I(tr) ⇒
. . | . |  ¬tr = nil ⇒
. . | . |  (∃L_1,L_2:|E| List.
. . | . |   tr = (L_1 @ L_2)
. . | . |   & ¬L_2 = nil
. . | . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | . 9. n : ℤ
. . | . 10. 0<n
. . | . 11. ∀tr:|E| List.
. . | . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | . 12. tr : |E| List
. . | . 13. ||tr||≤n
. . | . 14. ¬tr = nil
. . | . 15. ∀m:Label. P(<tr>_m)
. . | . 16. I(tr)
. . | . 17. L_1 : |E| List
. . | . 18. L_2 : |E| List
. . | . 19. tr = (L_1 @ L_2)
. . | . 20. ¬L_2 = nil
. . | . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . 22. m : Label
. . | . 23. (∀x∈L_2.tag(E)(x) = m)
. . | . 24. ||L_1||≤n-1
. . | . 25. L_1 ≤ tr
. . | . 26. I(L_1)
. . | . ⊢  P(L_1)  by Assert ||L_1||≤n-1
. . | . |
. . | . |\
. . | . | .....assertion .....
. . | . | 1. E : TaggedEventStruct
. . | . | 2. P : (|E| List)→Prop
. . | . | 3. I : (|E| List)→Prop
. . | . | 4. memorylessR(E) preserves P
. . | . | 5. ∀x,y,z:|E| List.
. . | . |    P(x) ⇒
. . | . |    P(y) ⇒
```

```
. . | . |    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | . | 6. safetyR(E) preserves P
. . | . | 7. safetyR(E) preserves I
. . | . | 8. ∀tr:|E| List.
. . | . |    I(tr) ⇒
. . | . |    ¬tr = nil ⇒
. . | . |    (∃L_1,L_2:|E| List.
. . | . |     tr = (L_1 @ L_2)
. . | . |     & ¬L_2 = nil
. . | . |     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . |     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | . | 9. n : ℤ
. . | . | 10. 0<n
. . | . | 11. ∀tr:|E| List.
. . | . |     ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | . | 12. tr : |E| List
. . | . | 13. ||tr||≤n
. . | . | 14. ¬tr = nil
. . | . | 15. ∀m:Label. P(<tr>_m)
. . | . | 16. I(tr)
. . | . | 17. L_1 : |E| List
. . | . | 18. L_2 : |E| List
. . | . | 19. tr = (L_1 @ L_2)
. . | . | 20. ¬L_2 = nil
. . | . | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . | 22. m : Label
. . | . | 23. (∀x∈L_2.tag(E)(x) = m)
. . | . | 24. ||L_1||≤n-1
. . | . | 25. L_1 ≤ tr
. . | . | 26. I(L_1)
. . | . | ⊢  ||L_1||≤n-1
. . | . |   by InstLemma `length_append` [|E|;L_1;L_2] THENA Auto
. . | . |
. . | . |      THEN InstLemma `non_nil_length` [|E|;L_2] THENA Auto
. . | . |
. . | . |      THEN SubstFor L_1 @ L_2 (-2) THEN Auto'
. . | . |
. . | . .\
. . | . . 1. E : TaggedEventStruct
. . | . . 2. P : (|E| List)→Prop
. . | . . 3. I : (|E| List)→Prop
. . | . . 4. memorylessR(E) preserves P
. . | . . 5. ∀x,y,z:|E| List.
. . | . . |  P(x) ⇒
. . | . . |  P(y) ⇒
. . | . . |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | . . 6. safetyR(E) preserves P
. . | . . 7. safetyR(E) preserves I
. . | . . 8. ∀tr:|E| List.
. . | . . |  I(tr) ⇒
. . | . . |  ¬tr = nil ⇒
. . | . . |  (∃L_1,L_2:|E| List.
. . | . . |   tr = (L_1 @ L_2)
. . | . . |   & ¬L_2 = nil
. . | . . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
```

```
. . | . . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | . . 9. n : ℤ
. . | . . 10. 0<n
. . | . . 11. ∀tr:|E| List.
. . | . . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | . . 12. tr : |E| List
. . | . . 13. ||tr||≤n
. . | . . 14. ¬tr = nil
. . | . . 15. ∀m:Label. P(<tr>_m)
. . | . . 16. I(tr)
. . | . . 17. L_1 : |E| List
. . | . . 18. L_2 : |E| List
. . | . . 19. tr = (L_1 @ L_2)
. . | . . 20. ¬L_2 = nil
. . | . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . . 22. m : Label
. . | . . 23. (∀x∈L_2.tag(E)(x) = m)
. . | . . 24. ||L_1||≤n-1
. . | . . 25. L_1 ≤ tr
. . | . . 26. I(L_1)
. . | . . 27. ||L_1||≤n-1
. . | . . ⊢  P(L_1)
. . | . . | by AssertBY L_1 ≤ tr (Unfold ‘iseg‘ 0 THEN InstConcl [L_2
. . | . . |    ] THEN Auto)
. . | . . THEN AssertBY I(L_1)
. . | . .
. . | . . |    ...
. . | . . |     THEN Using [‘x’,tr
. . | . . |    ] (BackThruHyp h) THEN Auto))
. . | . . |

. . | . . 1. E : TaggedEventStruct
. . | . . 2. P : (|E| List)→Prop
. . | . . 3. I : (|E| List)→Prop
. . | . . 4. memorylessR(E) preserves P
. . | . . 5. ∀x,y,z:|E| List.
. . | . . |  P(x) ⇒
. . | . . |  P(y) ⇒
. . | . . |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | . . 6. safetyR(E) preserves P
. . | . . 7. safetyR(E) preserves I
. . | . . 8. ∀tr:|E| List.
. . | . . |  I(tr) ⇒
. . | . . |  ¬tr = nil ⇒
. . | . . |  (∃L_1,L_2:|E| List.
. . | . . |   tr = (L_1 @ L_2)
. . | . . |   & ¬L_2 = nil
. . | . . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | . . 9. n : ℤ
. . | . . 10. 0<n
. . | . . 11. ∀tr:|E| List.
. . | . . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | . . 12. tr : |E| List
. . | . . 13. ||tr||≤n
. . | . . 14. ¬tr = nil
```

```
. . | . . 15. ∀m:Label. P(<tr>_m)
. . | . . 16. I(tr)
. . | . . 17. L_1 : |E| List
. . | . . 18. L_2 : |E| List
. . | . . 19. tr = (L_1 @ L_2)
. . | . . 20. ¬L_2 = nil
. . | . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . . 22. m : Label
. . | . . 23. (∀x∈L_2.tag(E)(x) = m)
. . | . . 24. ||L_1||≤n-1
. . | . . 25. L_1 ≤ tr
. . | . . 26. I(L_1)
. . | . . 27. ||L_1||≤n-1
. . | . . 28. L_1 ≤ tr
. . | . . 29. I(L_1)
. . | . . ⊢  P(L_1)
. . | . . | by AllHyps (\h. InstHyp [L_1
. . | . . |     ] h THENA Complete Auto THEN BackThruHyp (-1)
. . | . .
. . | . . |                        THEN Try (Complete Auto))
. . | . . |
. . | . . 1. E : TaggedEventStruct
. . | . . 2. P : (|E| List)→Prop
. . | . . 3. I : (|E| List)→Prop
. . | . . 4. memorylessR(E) preserves P
. . | . . 5. ∀x,y,z:|E| List.
. . | . . |  P(x) ⇒
. . | . . |  P(y) ⇒
. . | . . |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | . . 6. safetyR(E) preserves P
. . | . . 7. safetyR(E) preserves I
. . | . . 8. ∀tr:|E| List.
. . | . . |  I(tr) ⇒
. . | . . |  ¬tr = nil ⇒
. . | . . |  (∃L_1,L_2:|E| List.
. . | . . |   tr = (L_1 @ L_2)
. . | . . |   & ¬L_2 = nil
. . | . . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | . . 9. n : ℤ
. . | . . 10. 0<n
. . | . . 11. ∀tr:|E| List.
. . | . . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | . . 12. tr : |E| List
. . | . . 13. ||tr||≤n
. . | . . 14. ¬tr = nil
. . | . . 15. ∀m:Label. P(<tr>_m)
. . | . . 16. I(tr)
. . | . . 17. L_1 : |E| List
. . | . . 18. L_2 : |E| List
. . | . . 19. tr = (L_1 @ L_2)
. . | . . 20. ¬L_2 = nil
. . | . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . . 22. m : Label
. . | . . 23. (∀x∈L_2.tag(E)(x) = m)
```

```
. . | . . 24. ||L_1||≤n-1
. . | . . 25. L_1 ≤ tr
. . | . . 26. I(L_1)
. . | . . 27. ||L_1||≤n-1
. . | . . 28. L_1 ≤ tr
. . | . . 29. I(L_1)
. . | . . 30. (∀m:Label. P(<L_1>_m)) ⇒ I(L_1) ⇒ P(L_1)
. . | . . ⊢  ∀m:Label. P(<L_1>_m)
. . | . . | by AllHyps ParallelOp
. . | . .
. . | . . |    ...
. . | . . |                THEN Using [‘x’,<tr>_m1
. . | . . |    ] (BackThruHyp h THEN Auto))
. . | . . |
. . | . . 1. E : TaggedEventStruct
. . | . . 2. P : (|E| List)→Prop
. . | . . 3. I : (|E| List)→Prop
. . | . . 4. memorylessR(E) preserves P
. . | . . 5. ∀x,y,z:|E| List.
. . | . .    P(x) ⇒

. . | . .    P(y) ⇒
. . | . .    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . | . . 6. ∀x,y:|E| List. P(x) ⇒ y ≤ x ⇒ P(y)
. . | . . 7. safetyR(E) preserves I
. . | . . 8. ∀tr:|E| List.
. . | . .    I(tr) ⇒
. . | . .    ¬tr = nil ⇒
. . | . .    (∃L_1,L_2:|E| List.
. . | . .     tr = (L_1 @ L_2)
. . | . .     & ¬L_2 = nil
. . | . .     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . .     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . | . . 9. n : ℤ
. . | . . 10. 0<n
. . | . . 11. ∀tr:|E| List.
. . | . .     ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . | . . 12. tr : |E| List
. . | . . 13. ||tr||≤n
. . | . . 14. ¬tr = nil
. . | . . 15. ∀m:Label. P(<tr>_m)
. . | . . 16. I(tr)
. . | . . 17. L_1 : |E| List
. . | . . 18. L_2 : |E| List
. . | . . 19. tr = (L_1 @ L_2)
. . | . . 20. ¬L_2 = nil
. . | . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . | . . 22. m : Label
. . | . . 23. (∀x∈L_2.tag(E)(x) = m)
. . | . . 24. ||L_1||≤n-1
. . | . . 25. L_1 ≤ tr
. . | . . 26. I(L_1)
. . | . . 27. ||L_1||≤n-1
. . | . . 28. L_1 ≤ tr
. . | . . 29. I(L_1)
. . | . . 30. (∀m:Label. P(<L_1>_m)) ⇒ I(L_1) ⇒ P(L_1)
```

```
. . | . . 31. m1 : Label
. . | . . 32. P(<tr>_m1)
. . | . . ⊢  <L_1>_m1 ≤ <tr>_m1
. . | . .   by ...
. . | . .
. . |
. . .\
. . . .....antecedent .....
. . . 1. E : TaggedEventStruct
. . . 2. P : (|E| List)→Prop
. . . 3. I : (|E| List)→Prop
. . . 4. memorylessR(E) preserves P
. . . 5. ∀x,y,z:|E| List.
. . . |  P(x) ⇒
. . . |  P(y) ⇒ (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . 6. safetyR(E) preserves P
. . . 7. safetyR(E) preserves I
. . . 8. ∀tr:|E| List.
. . . |  I(tr) ⇒
. . . |  ¬tr = nil ⇒
. . . |  (∃L_1,L_2:|E| List.
. . . |   tr = (L_1 @ L_2)
. . . |   & ¬L_2 = nil
. . . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . 9. n : ℤ
. . . 10. 0<n
. . . 11. ∀tr:|E| List.
. . . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . 12. tr : |E| List
. . . 13. ||tr||≤n
. . . 14. ¬tr = nil
. . . 15. ∀m:Label. P(<tr>_m)
. . . 16. I(tr)
. . . 17. L_1 : |E| List
. . . 18. L_2 : |E| List
. . . 19. tr = (L_1 @ L_2)
. . . 20. ¬L_2 = nil
. . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . 22. m : Label
. . . 23. (∀x∈L_2.tag(E)(x) = m)
. . . ⊢  P(L_2)
. . . | by Subst L_2 = (<tr>_m -x =msg=(E) y L_1) 0 THENA Auto
. . . |
. . . |\
. . . | .....equality .....
. . . | 1. E : TaggedEventStruct
. . . | 2. P : (|E| List)→Prop
. . . | 3. I : (|E| List)→Prop
. . . | 4. memorylessR(E) preserves P
. . . | 5. ∀x,y,z:|E| List.
. . . | |  P(x) ⇒
. . . | |  P(y) ⇒ (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | 6. safetyR(E) preserves P
. . . | 7. safetyR(E) preserves I
```

```
. . . | 8. ∀tr:|E| List.
. . . | |  I(tr) ⇒
. . . | |  ¬tr = nil ⇒
. . . | |  (∃L_1,L_2:|E| List.
. . . | |   tr = (L_1 @ L_2)
. . . | |   & ¬L_2 = nil
. . . | |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | 9. n : ℤ
. . . | 10. 0<n
. . . | 11. ∀tr:|E| List.
. . . | |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | 12. tr : |E| List
. . . | 13. ||tr||≤n
. . . | 14. ¬tr = nil
. . . | 15. ∀m:Label. P(<tr>_m)
. . . | 16. I(tr)
. . . | 17. L_1 : |E| List
. . . | 18. L_2 : |E| List
. . . | 19. tr = (L_1 @ L_2)
. . . | 20. ¬L_2 = nil
. . . | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | 22. m : Label
. . . | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | ⊢  L_2 = (<tr>_m -x =msg=(E) y L_1)  by Assert <L_2>_m ∼ L_2
. . . | |
. . . | |\
. . . | | .....assertion .....
. . . | | 1. E : TaggedEventStruct
. . . | | 2. P : (|E| List)→Prop
. . . | | 3. I : (|E| List)→Prop
. . . | | 4. memorylessR(E) preserves P
. . . | | 5. ∀x,y,z:|E| List.
. . . | | |  P(x) ⇒
. . . | | |  P(y) ⇒
. . . | | |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | | 6. safetyR(E) preserves P
. . . | | 7. safetyR(E) preserves I
. . . | | 8. ∀tr:|E| List.
. . . | | |  I(tr) ⇒
. . . | | |  ¬tr = nil ⇒
. . . | | |  (∃L_1,L_2:|E| List.
. . . | | |   tr = (L_1 @ L_2)
. . . | | |   & ¬L_2 = nil
. . . | | |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | | |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | | 9. n : ℤ
. . . | | 10. 0<n
. . . | | 11. ∀tr:|E| List.
. . . | | |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | | 12. tr : |E| List
. . . | | 13. ||tr||≤n
. . . | | 14. ¬tr = nil
. . . | | 15. ∀m:Label. P(<tr>_m)
. . . | | 16. I(tr)
```

```
. . . | | 17. L_1 : |E| List
. . . | | 18. L_2 : |E| List
. . . | | 19. tr = (L_1 @ L_2)
. . . | | 20. ¬L_2 = nil
. . . | | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | | 22. m : Label
. . . | | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | | ⊢  <L_2>_m ∼ L_2
. . . | | | by ...
. . . | | |
. . . | | 1. E : TaggedEventStruct
. . . | | 2. P : (|E| List)→Prop
. . . | | 3. I : (|E| List)→Prop
. . . | | 4. memorylessR(E) preserves P
. . . | | 5. ∀x,y,z:|E| List.
. . . | |    P(x) ⇒
. . . | |    P(y) ⇒
. . . | |    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | | 6. safetyR(E) preserves P
. . . | | 7. safetyR(E) preserves I
. . . | | 8. ∀tr:|E| List.
. . . | |    I(tr) ⇒
. . . | |    ¬tr = nil ⇒
. . . | |    (∃L_1,L_2:|E| List.
. . . | |     tr = (L_1 @ L_2)
. . . | |     & ¬L_2 = nil
. . . | |     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | |     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | | 9. n : ℤ
. . . | | 10. 0<n
. . . | | 11. ∀tr:|E| List.
. . . | |     ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | | 12. tr : |E| List
. . . | | 13. ||tr||≤n
. . . | | 14. ¬tr = nil
. . . | | 15. ∀m:Label. P(<tr>_m)
. . . | | 16. I(tr)
. . . | | 17. L_1 : |E| List
. . . | | 18. L_2 : |E| List
. . . | | 19. tr = (L_1 @ L_2)
. . . | | 20. ¬L_2 = nil
. . . | | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | | 22. m : Label
. . . | | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | | ⊢  (∀x∈L_2.(λe.tag(E)(e) =₁ m)(x))
. . . | |   by RepeatFor 2 (ParallelOp (-1))
. . . | |
. . . | |      THEN Unfold `so_apply` 0 THEN Reduce 0
. . . | |
. . . | |      THEN ParallelOp (-1)
. . . | | THEN HypSubstSq (-1) 0
. . . | |
. . . | |      THEN EqLblReflexive 0
. . . | |
. . . | .\
```

```
. . . | . 1. E : TaggedEventStruct
. . . | . 2. P : (|E| List)→Prop
. . . | . 3. I : (|E| List)→Prop
. . . | . 4. memorylessR(E) preserves P
. . . | . 5. ∀x,y,z:|E| List.
. . . | . |  P(x) ⇒
. . . | . |  P(y) ⇒
. . . | . |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . 6. safetyR(E) preserves P
. . . | . 7. safetyR(E) preserves I
. . . | . 8. ∀tr:|E| List.
. . . | . |  I(tr) ⇒
. . . | . |  ¬tr = nil ⇒
. . . | . |  (∃L_1,L_2:|E| List.
. . . | . |   tr = (L_1 @ L_2)
. . . | . |   & ¬L_2 = nil
. . . | . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . 9. n : ℤ
. . . | . 10. 0<n
. . . | . 11. ∀tr:|E| List.
. . . | . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . 12. tr : |E| List
. . . | . 13. ||tr||≤n
. . . | . 14. ¬tr = nil
. . . | . 15. ∀m:Label. P(<tr>_m)
. . . | . 16. I(tr)
. . . | . 17. L_1 : |E| List
. . . | . 18. L_2 : |E| List
. . . | . 19. tr = (L_1 @ L_2)
. . . | . 20. ¬L_2 = nil
. . . | . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . 22. m : Label
. . . | . 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . 24. <L_2>_m ∼ L_2
. . . | . ⊢  L_2 = (<tr>_m -x =msg=(E) y L_1)
. . . | . | by Assert L_2 = (<L_2>_m -x =msg=(E) y L_1)
. . . | . |
. . . | . |\
. . . | . | .....assertion .....
. . . | . | 1. E : TaggedEventStruct
. . . | . | 2. P : (|E| List)→Prop
. . . | . | 3. I : (|E| List)→Prop
. . . | . | 4. memorylessR(E) preserves P
. . . | . | 5. ∀x,y,z:|E| List.
. . . | . | |  P(x) ⇒
. . . | . | |  P(y) ⇒
. . . | . | |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . | 6. safetyR(E) preserves P
. . . | . | 7. safetyR(E) preserves I
. . . | . | 8. ∀tr:|E| List.
. . . | . | |  I(tr) ⇒
. . . | . | |  ¬tr = nil ⇒
. . . | . | |  (∃L_1,L_2:|E| List.
. . . | . | |   tr = (L_1 @ L_2)
```

```
. . . | . | |   & ¬L_2 = nil
. . . | . | |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . | 9. n : ℤ
. . . | . | 10. 0<n
. . . | . | 11. ∀tr:|E| List.
. . . | . | |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . | 12. tr : |E| List
. . . | . | 13. ||tr||≤n
. . . | . | 14. ¬tr = nil
. . . | . | 15. ∀m:Label. P(<tr>_m)
. . . | . | 16. I(tr)
. . . | . | 17. L_1 : |E| List
. . . | . | 18. L_2 : |E| List
. . . | . | 19. tr = (L_1 @ L_2)
. . . | . | 20. ¬L_2 = nil
. . . | . | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | 22. m : Label
. . . | . | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . | 24. <L_2>_m ∼ L_2
. . . | . | ⊢  L_2 = (<L_2>_m -x =msg=(E) y L_1)
. . . | . | | by RWO "remove_msgs_disjoint" 0 THENA Auto
. . . | . | |
. . . | . | |\
. . . | . | | 1. E : TaggedEventStruct
. . . | . | | 2. P : (|E| List)→Prop
. . . | . | | 3. I : (|E| List)→Prop
. . . | . | | 4. memorylessR(E) preserves P
. . . | . | | 5. ∀x,y,z:|E| List.
. . . | . | | |  P(x) ⇒
. . . | . | | |  P(y) ⇒
. . . | . | | |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . | | 6. safetyR(E) preserves P
. . . | . | | 7. safetyR(E) preserves I
. . . | . | | 8. ∀tr:|E| List.
. . . | . | | |  I(tr) ⇒
. . . | . | | |  ¬tr = nil ⇒
. . . | . | | |  (∃L_1,L_2:|E| List.
. . . | . | | |   tr = (L_1 @ L_2)
. . . | . | | |   & ¬L_2 = nil
. . . | . | | |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | | |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . | | 9. n : ℤ
. . . | . | | 10. 0<n
. . . | . | | 11. ∀tr:|E| List.
. . . | . | | |   ||tr||≤n-1 ⇒
. . . | . | | |   (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . | | 12. tr : |E| List
. . . | . | | 13. ||tr||≤n
. . . | . | | 14. ¬tr = nil
. . . | . | | 15. ∀m:Label. P(<tr>_m)
. . . | . | | 16. I(tr)
. . . | . | | 17. L_1 : |E| List
. . . | . | | 18. L_2 : |E| List
. . . | . | | 19. tr = (L_1 @ L_2)
```

```
. . . | . | | 20. ¬L_2 = nil
. . . | . | | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | | 22. m : Label
. . . | . | | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . | | 24. <L_2>_m ∼ L_2
. . . | . | | 25. x : |E|
. . . | . | | 26. y : |E|
. . . | . | | 27. (x ∈ <L_2>_m)
. . . | . | | 28. (y ∈ L_1)
. . . | . | | ⊢  ¬(x =msg=(E) y)
. . . | . | | | by AssertBY ¬(y =msg=(E) x)
. . . | . | |
. . . | . | | |    ...
. . . | . | |
. . . | . | | |    ...
. . . | . | | |
. . . | . | | 1. E : TaggedEventStruct
. . . | . | | 2. P : (|E| List)→Prop
. . . | . | | 3. I : (|E| List)→Prop
. . . | . | | 4. memorylessR(E) preserves P
. . . | . | | 5. ∀x,y,z:|E| List.
. . . | . | |    P(x) ⇒
. . . | . | |    P(y) ⇒
. . . | . | |    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . | | 6. safetyR(E) preserves P
. . . | . | | 7. safetyR(E) preserves I
. . . | . | | 8. ∀tr:|E| List.
. . . | . | |    I(tr) ⇒
. . . | . | |    ¬tr = nil ⇒
. . . | . | |    (∃L_1,L_2:|E| List.
. . . | . | |     tr = (L_1 @ L_2)
. . . | . | |     & ¬L_2 = nil
. . . | . | |     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | |     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . | | 9. n : ℤ
. . . | . | | 10. 0<n
. . . | . | | 11. ∀tr:|E| List.
. . . | . | |     ||tr||≤n-1 ⇒
. . . | . | |     (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . | | 12. tr : |E| List
. . . | . | | 13. ||tr||≤n
. . . | . | | 14. ¬tr = nil
. . . | . | | 15. ∀m:Label. P(<tr>_m)
. . . | . | | 16. I(tr)
. . . | . | | 17. L_1 : |E| List
. . . | . | | 18. L_2 : |E| List
. . . | . | | 19. tr = (L_1 @ L_2)
. . . | . | | 20. ¬L_2 = nil
. . . | . | | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | | 22. m : Label
. . . | . | | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . | | 24. <L_2>_m ∼ L_2
. . . | . | | 25. x : |E|
. . . | . | | 26. y : |E|
. . . | . | | 27. (x ∈ <L_2>_m)
```

```
. . . | . | | 28. (y ∈ L_1)
. . . | . | | 29. ¬(y =msg=(E) x)
. . . | . | | ⊢  ¬(x =msg=(E) y)
. . . | . | |   by ...E]
. . . | . | | THENA Auto THEN UseEquiv
. . . | . | |
. . . | . | .\
. . . | . | . 1. E : TaggedEventStruct
. . . | . | . 2. P : (|E| List)→Prop
. . . | . | . 3. I : (|E| List)→Prop
. . . | . | . 4. memorylessR(E) preserves P
. . . | . | . 5. ∀x,y,z:|E| List.
. . . | . | .    P(x) ⇒
. . . | . | .    P(y) ⇒
. . . | . | .    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . | . 6. safetyR(E) preserves P
. . . | . | . 7. safetyR(E) preserves I
. . . | . | . 8. ∀tr:|E| List.
. . . | . | .    I(tr) ⇒
. . . | . | .    ¬tr = nil ⇒
. . . | . | .    (∃L_1,L_2:|E| List.
. . . | . | .     tr = (L_1 @ L_2)
. . . | . | .     & ¬L_2 = nil
. . . | . | .     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | .     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . | . 9. n : ℤ
. . . | . | . 10. 0<n
. . . | . | . 11. ∀tr:|E| List.
. . . | . | .     ||tr||≤n-1 ⇒
. . . | . | .     (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . | . 12. tr : |E| List
. . . | . | . 13. ||tr||≤n
. . . | . | . 14. ¬tr = nil
. . . | . | . 15. ∀m:Label. P(<tr>_m)
. . . | . | . 16. I(tr)
. . . | . | . 17. L_1 : |E| List
. . . | . | . 18. L_2 : |E| List
. . . | . | . 19. tr = (L_1 @ L_2)
. . . | . | . 20. ¬L_2 = nil
. . . | . | . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . | . 22. m : Label
. . . | . | . 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . | . 24. <L_2>_m ~ L_2
. . . | . | . ⊢  L_2 = <L_2>_m  by HypSubst (-1) 0 THEN Auto
. . . | . |
. . . | . .\
. . . | . . 1. E : TaggedEventStruct
. . . | . . 2. P : (|E| List)→Prop
. . . | . . 3. I : (|E| List)→Prop
. . . | . . 4. memorylessR(E) preserves P
. . . | . . 5. ∀x,y,z:|E| List.
. . . | . . |  P(x) ⇒
. . . | . . |  P(y) ⇒
. . . | . . |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . . 6. safetyR(E) preserves P
```

```
. . . | . . 7. safetyR(E) preserves I
. . . | . . 8. ∀tr:|E| List.
. . . | . . |  I(tr) ⇒
. . . | . . |  ¬tr = nil ⇒
. . . | . . |  (∃L_1,L_2:|E| List.
. . . | . . |   tr = (L_1 @ L_2)
. . . | . . |   & ¬L_2 = nil
. . . | . . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . . 9. n : ℤ
. . . | . . 10. 0<n
. . . | . . 11. ∀tr:|E| List.
. . . | . . |   ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . . 12. tr : |E| List
. . . | . . 13. ||tr||≤n
. . . | . . 14. ¬tr = nil
. . . | . . 15. ∀m:Label. P(<tr>_m)
. . . | . . 16. I(tr)
. . . | . . 17. L_1 : |E| List
. . . | . . 18. L_2 : |E| List
. . . | . . 19. tr = (L_1 @ L_2)
. . . | . . 20. ¬L_2 = nil
. . . | . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . 22. m : Label
. . . | . . 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . . 24. <L_2>_m ∼ L_2
. . . | . . 25. L_2 = (<L_2>_m -x =msg=(E) y L_1)
. . . | . . ⊢  L_2 = (<tr>_m -x =msg=(E) y L_1)
. . . | . . | by Assert (<L_1>_m -x =msg=(E) y L_1) = nil
. . . | . . |
. . . | . . |\
. . . | . . | .....assertion .....
. . . | . . | 1. E : TaggedEventStruct
. . . | . . | 2. P : (|E| List)→Prop
. . . | . . | 3. I : (|E| List)→Prop
. . . | . . | 4. memorylessR(E) preserves P
. . . | . . | 5. ∀x,y,z:|E| List.
. . . | . . | |  P(x) ⇒
. . . | . . | |  P(y) ⇒
. . . | . . | |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . . | 6. safetyR(E) preserves P
. . . | . . | 7. safetyR(E) preserves I
. . . | . . | 8. ∀tr:|E| List.
. . . | . . | |  I(tr) ⇒
. . . | . . | |  ¬tr = nil ⇒
. . . | . . | |  (∃L_1,L_2:|E| List.
. . . | . . | |   tr = (L_1 @ L_2)
. . . | . . | |   & ¬L_2 = nil
. . . | . . | |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . | |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . . | 9. n : ℤ
. . . | . . | 10. 0<n
. . . | . . | 11. ∀tr:|E| List.
. . . | . . | |   ||tr||≤n-1 ⇒
. . . | . . | |   (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
```

```
. . . | . . | 12. tr : |E| List
. . . | . . | 13. ||tr||≤n
. . . | . . | 14. ¬tr = nil
. . . | . . | 15. ∀m:Label. P(<tr>_m)
. . . | . . | 16. I(tr)
. . . | . . | 17. L_1 : |E| List
. . . | . . | 18. L_2 : |E| List
. . . | . . | 19. tr = (L_1 @ L_2)
. . . | . . | 20. ¬L_2 = nil
. . . | . . | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . | 22. m : Label
. . . | . . | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . . | 24. <L_2>_m ∼ L_2
. . . | . . | 25. L_2 = (<L_2>_m -x =msg=(E) y L_1)
. . . | . . | ⊢  (<L_1>_m -x =msg=(E) y L_1) = nil
. . . | . . | | by BackThruLemma ‘remove_msgs_nil‘ THEN Auto
. . . | . . | |
. . . | . . | |\
. . . | . . | | 1. E : TaggedEventStruct
. . . | . . | | 2. P : (|E| List)→Prop
. . . | . . | | 3. I : (|E| List)→Prop
. . . | . . | | 4. memorylessR(E) preserves P
. . . | . . | | 5. ∀x,y,z:|E| List.
. . . | . . | |    P(x) ⇒
. . . | . . | |    P(y) ⇒
. . . | . . | |    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . . | | 6. safetyR(E) preserves P
. . . | . . | | 7. safetyR(E) preserves I
. . . | . . | | 8. ∀tr:|E| List.
. . . | . . | |    I(tr) ⇒
. . . | . . | |    ¬tr = nil ⇒
. . . | . . | |    (∃L_1,L_2:|E| List.
. . . | . . | |     tr = (L_1 @ L_2)
. . . | . . | |     & ¬L_2 = nil
. . . | . . | |     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . | |     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . . | | 9. n : ℤ
. . . | . . | | 10. 0<n
. . . | . . | | 11. ∀tr:|E| List.
. . . | . . | |     ||tr||≤n-1 ⇒
. . . | . . | |     (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . . | | 12. tr : |E| List
. . . | . . | | 13. ||tr||≤n
. . . | . . | | 14. ¬tr = nil
. . . | . . | | 15. ∀m:Label. P(<tr>_m)
. . . | . . | | 16. I(tr)
. . . | . . | | 17. L_1 : |E| List
. . . | . . | | 18. L_2 : |E| List
. . . | . . | | 19. tr = (L_1 @ L_2)
. . . | . . | | 20. ¬L_2 = nil
. . . | . . | | 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . | | 22. m : Label
. . . | . . | | 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . . | | 24. <L_2>_m ∼ L_2
. . . | . . | | 25. L_2 = (<L_2>_m -x =msg=(E) y L_1)
```

```
. . . | . . | | 26. x : |E|
. . . | . . | | 27. (x ∈ <L_1>_m)
. . . | . . | | ⊢  (x ∈ L_1)
. . . | . . | |   by Unfold ‘tag_sublist‘ (-1) THEN
. . . | . . | |
. . . | . . | |      RWO "member_filter" (-1) THEN Auto
. . . | . . | |
. . . | . . | .\
. . . | . . | . 1. E : TaggedEventStruct
. . . | . . | . 2. P : (|E| List)→Prop
. . . | . . | . 3. I : (|E| List)→Prop
. . . | . . | . 4. memorylessR(E) preserves P
. . . | . . | . 5. ∀x,y,z:|E| List.
. . . | . . | .    P(x) ⇒
. . . | . . | .    P(y) ⇒
. . . | . . | .    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . . | . 6. safetyR(E) preserves P
. . . | . . | . 7. safetyR(E) preserves I
. . . | . . | . 8. ∀tr:|E| List.
. . . | . . | .    I(tr) ⇒
. . . | . . | .    ¬tr = nil ⇒
. . . | . . | .    (∃L_1,L_2:|E| List.
. . . | . . | .     tr = (L_1 @ L_2)
. . . | . . | .     & ¬L_2 = nil
. . . | . . | .     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . | .     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . . | . 9. n : ℤ
. . . | . . | . 10. 0<n
. . . | . . | . 11. ∀tr:|E| List.
. . . | . . | .     ||tr||≤n-1 ⇒
. . . | . . | .     (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . . | . 12. tr : |E| List
. . . | . . | . 13. ||tr||≤n
. . . | . . | . 14. ¬tr = nil
. . . | . . | . 15. ∀m:Label. P(<tr>_m)
. . . | . . | . 16. I(tr)
. . . | . . | . 17. L_1 : |E| List
. . . | . . | . 18. L_2 : |E| List
. . . | . . | . 19. tr = (L_1 @ L_2)
. . . | . . | . 20. ¬L_2 = nil
. . . | . . | . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . | . 22. m : Label
. . . | . . | . 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . . | . 24. <L_2>_m ∼ L_2
. . . | . . | . 25. L_2 = (<L_2>_m -x =msg=(E) y L_1)
. . . | . . | . ⊢  Refl(|E|)(_1 =msg=(E) _2)
. . . | . . | .   by InstLemma ‘event_msg_eq_equiv‘ [E] THENA Auto
. . . | . . | .
. . . | . . | .      THEN Unfold ‘equiv_rel‘ (-1) THEN Auto
. . . | . . |
. . . | . . .\
. . . | . . . 1. E : TaggedEventStruct
. . . | . . . 2. P : (|E| List)→Prop
. . . | . . . 3. I : (|E| List)→Prop
. . . | . . . 4. memorylessR(E) preserves P
```

```
. . . | . . . 5. ∀x,y,z:|E| List.
. . . | . . . |  P(x) ⇒
. . . | . . . |  P(y) ⇒
. . . | . . . |  (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . . . 6. safetyR(E) preserves P
. . . | . . . 7. safetyR(E) preserves I
. . . | . . . 8. ∀tr:|E| List.
. . . | . . . |  I(tr) ⇒
. . . | . . . |  ¬tr = nil ⇒
. . . | . . . |  (∃L_1,L_2:|E| List.
. . . | . . . |   tr = (L_1 @ L_2)
. . . | . . . |   & ¬L_2 = nil
. . . | . . . |   & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . . |   & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . . . 9. n : ℤ
. . . | . . . 10. 0<n
. . . | . . . 11. ∀tr:|E| List.
. . . | . . . |   ||tr||≤n-1 ⇒
. . . | . . . |   (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . . . 12. tr : |E| List
. . . | . . . 13. ||tr||≤n
. . . | . . . 14. ¬tr = nil
. . . | . . . 15. ∀m:Label. P(<tr>_m)
. . . | . . . 16. I(tr)
. . . | . . . 17. L_1 : |E| List
. . . | . . . 18. L_2 : |E| List
. . . | . . . 19. tr = (L_1 @ L_2)
. . . | . . . 20. ¬L_2 = nil
. . . | . . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . . 22. m : Label
. . . | . . . 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . . . 24. <L_2>_m ∼ L_2
. . . | . . . 25. L_2 = (<L_2>_m -x =msg=(E) y L_1)
. . . | . . . 26. (<L_1>_m -x =msg=(E) y L_1) = nil
. . . | . . . ⊢  L_2 = (<tr>_m -x =msg=(E) y L_1)
. . . | . . . | by SubstFor tr 0
. . . | . . . |  THEN Unfold ‘tag_sublist‘ 0
. . . | . . . |
. . . | . . . |  THEN RW FilterAppendC 0 THENA Auto
. . . | . . .
. . . | . . . |       THEN Fold ‘tag_sublist‘ 0
. . . | . . .
. . . | . . . |       THEN Unfold ‘remove_msgs‘ 0
. . . | . . .
. . . | . . . |       THEN RW FilterAppendC 0 THENA Auto
. . . | . . .
. . . | . . . |       THEN Fold ‘remove_msgs‘ 0
. . . | . . . |
. . . | . . . 1. E : TaggedEventStruct
. . . | . . . 2. P : (|E| List)→Prop
. . . | . . . 3. I : (|E| List)→Prop
. . . | . . . 4. memorylessR(E) preserves P
. . . | . . . 5. ∀x,y,z:|E| List.
. . . | . . .    P(x) ⇒
. . . | . . .    P(y) ⇒
```

```
. . . | . . .    (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . | . . . 6. safetyR(E) preserves P
. . . | . . . 7. safetyR(E) preserves I
. . . | . . . 8. ∀tr:|E| List.
. . . | . . .    I(tr) ⇒
. . . | . . .    ¬tr = nil ⇒
. . . | . . .    (∃L_1,L_2:|E| List.
. . . | . . .     tr = (L_1 @ L_2)
. . . | . . .     & ¬L_2 = nil
. . . | . . .     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . .     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
. . . | . . . 9. n : ℤ
. . . | . . . 10. 0<n
. . . | . . . 11. ∀tr:|E| List.
. . . | . . .     ||tr||≤n-1 ⇒
. . . | . . .     (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . | . . . 12. tr : |E| List
. . . | . . . 13. ||tr||≤n
. . . | . . . 14. ¬tr = nil
. . . | . . . 15. ∀m:Label. P(<tr>_m)
. . . | . . . 16. I(tr)
. . . | . . . 17. L_1 : |E| List
. . . | . . . 18. L_2 : |E| List
. . . | . . . 19. tr = (L_1 @ L_2)
. . . | . . . 20. ¬L_2 = nil
. . . | . . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . | . . . 22. m : Label
. . . | . . . 23. (∀x∈L_2.tag(E)(x) = m)
. . . | . . . 24. <L_2>_m ~ L_2
. . . | . . . 25. L_2 = (<L_2>_m -x =msg=(E) y L_1)
. . . | . . . 26. (<L_1>_m -x =msg=(E) y L_1) = nil
. . . | . . . ⊢  L_2
. . . | . . .    =
. . . | . . .    ((<L_1>_m -x =msg=(E) y L_1)
. . . | . . .     @ (<L_2>_m -x =msg=(E) y L_1))
. . . | . . .   by HypSubst (-1) 0 THEN Reduce 0 THEN Auto
. . . |
. . . .\
. . . . 1. E : TaggedEventStruct
. . . . 2. P : (|E| List)→Prop
. . . . 3. I : (|E| List)→Prop
. . . . 4. memorylessR(E) preserves P
. . . . 5. ∀x,y,z:|E| List.
. . . .    P(x) ⇒
. . . .    P(y) ⇒ (∀x∈x.(∀y∈y.¬(x =msg=(E) y))) & z = (x @ y) ⇒ P(z)
. . . . 6. safetyR(E) preserves P
. . . . 7. safetyR(E) preserves I
. . . . 8. ∀tr:|E| List.
. . . .    I(tr) ⇒
. . . .    ¬tr = nil ⇒
. . . .    (∃L_1,L_2:|E| List.
. . . .     tr = (L_1 @ L_2)
. . . .     & ¬L_2 = nil
. . . .     & (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . .     & (∃m:Label. (∀x∈L_2.tag(E)(x) = m)))
```

```
. . . . 9. n : ℤ
. . . . 10. 0<n
. . . . 11. ∀tr:|E| List.
. . . .     ||tr||≤n-1 ⇒ (∀m:Label. P(<tr>_m)) ⇒ I(tr) ⇒ P(tr)
. . . . 12. tr : |E| List

. . . . 13. ||tr||≤n
. . . . 14. ¬tr = nil
. . . . 15. ∀m:Label. P(<tr>_m)
. . . . 16. I(tr)
. . . . 17. L_1 : |E| List
. . . . 18. L_2 : |E| List
. . . . 19. tr = (L_1 @ L_2)
. . . . 20. ¬L_2 = nil
. . . . 21. (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . . 22. m : Label
. . . . 23. (∀x∈L_2.tag(E)(x) = m)
. . . . ⊢  P((<tr>_m -x =msg=(E) y L_1))
. . . .   by BackThruLemma ‘memoryless_remove_msgs‘ THEN Auto
. . . .
. . . .      THEN Try (Unfold ‘trace_property‘ 0 THEN Auto)
. . . .
. . . .      THEN EasyHyp
END
```