```
switch_decomp_implies_single_tag_decomp  THM

⊢  ∀E:TaggedEventStruct.
|  (switch-decomposable(E)
|  ∧ Tag-by-msg(E)
|  ∧ Causal(E)
|  ∧ No-dup-send(E))
|  refines single-tag-decomposable(E)
| by Auto
THEN Unfolds ``P_tag_by_msg prop_and`` 0

|    ...
THEN Reduce 0 THEN Auto

|    THEN SplitOrHyps THEN Try Trivial THEN ExRepD

|    THEN InstLemma `append_split2` [|E|;tr;C(Q)
|    ] THENA Try (Complete (Auto THEN Reduce 0 THEN Auto))
|
|
|\
| .....antecedent .....
| 1. E : TaggedEventStruct
| 2. tr : |E| List
| 3. Q : ℕ||tr||→Prop
| 4. ∀i:ℕ||tr||. Dec(Q(i))
| 5. i : ℕ||tr||
| 6. Q(i)
| 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
| 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
| 10. ∀i,j:ℕ||tr||.
|     (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 11. Causal(E)(tr)
| 12. No-dup-send(E)(tr)
| 13. ¬tr = nil
| ⊢  ∀x:ℕ||tr||. Dec(C(Q)(x))
|   by Unfold `message_closure` 0 THEN Reduce 0 THEN Auto
|
|\
| .....antecedent .....
| 1. E : TaggedEventStruct
| 2. tr : |E| List
| 3. Q : ℕ||tr||→Prop
| 4. ∀i:ℕ||tr||. Dec(Q(i))
| 5. i : ℕ||tr||
| 6. Q(i)
| 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
| 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
| 10. ∀i,j:ℕ||tr||.
| |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 11. Causal(E)(tr)
| 12. No-dup-send(E)(tr)
| 13. ¬tr = nil
```

```
| ⊢  ∀i,j:ℕ||tr||. C(Q)(i) ⇒ i<j ⇒ C(Q)(j)
| | by ...
| |
| 1. E : TaggedEventStruct
| 2. tr : |E| List
| 3. Q : ℕ||tr||→Prop
| 4. ∀i:ℕ||tr||. Dec(Q(i))
| 5. i : ℕ||tr||
| 6. Q(i)
| 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
| 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
| 10. ∀i,j:ℕ||tr||.
| |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 11. Causal(E)(tr)
| 12. No-dup-send(E)(tr)
| 13. ¬tr = nil
| 14. i1 : ℕ||tr||
| 15. j : ℕ||tr||
| 16. k : ℕ||tr||
| 17. Q(k)
| 18. tr[k] =msg=(E) tr[i1]
| 19. i1<j
| ⊢  C(Q)(j)  by Assert k≤j THEN EasyHyp
| |
| 1. E : TaggedEventStruct
| 2. tr : |E| List
| 3. Q : ℕ||tr||→Prop
| 4. ∀i:ℕ||tr||. Dec(Q(i))
| 5. i : ℕ||tr||
| 6. Q(i)
| 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
| 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
| 10. ∀i,j:ℕ||tr||.
| |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 11. Causal(E)(tr)
| 12. No-dup-send(E)(tr)
| 13. ¬tr = nil
| 14. i1 : ℕ||tr||
| 15. j : ℕ||tr||
| 16. k : ℕ||tr||
| 17. Q(k)
| 18. tr[k] =msg=(E) tr[i1]
| 19. i1<j
| ⊢  k≤j  by Assert k≤i1 THEN Auto'
| |
| .....assertion .....
| 1. E : TaggedEventStruct
| 2. tr : |E| List
| 3. Q : ℕ||tr||→Prop
| 4. ∀i:ℕ||tr||. Dec(Q(i))
| 5. i : ℕ||tr||
| 6. Q(i)
| 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
```

```
| 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
| 10. ∀i,j:ℕ||tr||.
| |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 11. Causal(E)(tr)
| 12. No-dup-send(E)(tr)

| 13. ¬tr = nil
| 14. i1 : ℕ||tr||
| 15. j : ℕ||tr||
| 16. k : ℕ||tr||
| 17. Q(k)
| 18. tr[k] =msg=(E) tr[i1]
| 19. i1<j
| ⊢  k≤i1
| | by AllHyps (\h.Unfold `P_causal` h THEN Reduce h
|
| |                    THEN InstHyp [i1] h THENA Auto THEN ExRepD)
|
| |    THEN AllHyps (\h. Unfold `no_duplicate_send` h THEN Reduce h
|
| |                      THEN InstHyp [j1;k
| |    ] h THEN Auto THENA EasyHyp)
| |
| |
| 1. E : TaggedEventStruct
| 2. tr : |E| List
| 3. Q : ℕ||tr||→Prop
| 4. ∀i:ℕ||tr||. Dec(Q(i))
| 5. i : ℕ||tr||
| 6. Q(i)
| 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
| 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
| 10. ∀i,j:ℕ||tr||.
|     (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
| 11. ∀i:ℕ||tr||.
|     ∃j:ℕ||tr||. j≤i & is-send(E)(tr[j]) & (tr[j] =msg=(E) tr[i])
| 12. ∀i,j:ℕ||tr||.
|     is-send(E)(tr[i]) ⇒
|     is-send(E)(tr[j]) ⇒ (tr[i] =msg=(E) tr[j]) ⇒ i = j
| 13. ¬tr = nil
| 14. i1 : ℕ||tr||
| 15. j : ℕ||tr||
| 16. k : ℕ||tr||
| 17. Q(k)
| 18. tr[k] =msg=(E) tr[i1]
| 19. i1<j
| 20. j1 : ℕ||tr||
| 21. j1≤i1
| 22. is-send(E)(tr[j1])
| 23. tr[j1] =msg=(E) tr[i1]
| ⊢  tr[j1] =msg=(E) tr[k]
|   by InstLemma `event_msg_eq_equiv` [E] THENA Auto
| THEN UseTrans
|      tr[i1]
```

```
| THEN UseSym
|
.\
. 1. E : TaggedEventStruct
. 2. tr : |E| List
. 3. Q : ℕ||tr||→Prop
. 4. ∀i:ℕ||tr||. Dec(Q(i))
. 5. i : ℕ||tr||
. 6. Q(i)
. 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. 10. ∀i,j:ℕ||tr||.
. |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. 11. Causal(E)(tr)
. 12. No-dup-send(E)(tr)
. 13. ¬tr = nil
. 14. ∃L_1,L_2:|E| List.
. |   tr = (L_1 @ L_2) & (∀i:ℕ||tr||. C(Q)(i) ⟺ ||L_1||≤i)
. ⊢  ∃L_1,L_2:Trace(E).
. |  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))
. | by Unfold `message_closure` (-1) THEN Reduce (-1) THEN ExRepD
. |
. 1. E : TaggedEventStruct
. 2. tr : |E| List
. 3. Q : ℕ||tr||→Prop
. 4. ∀i:ℕ||tr||. Dec(Q(i))
. 5. i : ℕ||tr||
. 6. Q(i)
. 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. 10. ∀i,j:ℕ||tr||.
. |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. 11. Causal(E)(tr)
. 12. No-dup-send(E)(tr)
. 13. ¬tr = nil
. 14. L_1 : |E| List
. 15. L_2 : |E| List
. 16. tr = (L_1 @ L_2)
. 17. ∀i:ℕ||tr||.
. |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. ⊢  ∃L_1,L_2:Trace(E).
. |  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))
. | by Assert ||tr|| = ||L_1||+||L_2||

. |
. |
. |\
. | .....assertion .....
```

```
. | 1. E : TaggedEventStruct
. | 2. tr : |E| List
. | 3. Q : ℕ||tr||→Prop
. | 4. ∀i:ℕ||tr||. Dec(Q(i))
. | 5. i : ℕ||tr||
. | 6. Q(i)
. | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. | 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. | 10. ∀i,j:ℕ||tr||.
. |     (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. | 11. Causal(E)(tr)
. | 12. No-dup-send(E)(tr)
. | 13. ¬tr = nil
. | 14. L_1 : |E| List
. | 15. L_2 : |E| List
. | 16. tr = (L_1 @ L_2)
. | 17. ∀i:ℕ||tr||.
. |     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. | ⊢  ||tr|| = ||L_1||+||L_2||
. |   by SubstFor tr
. |       0 THENA Auto' THEN BackThruLemma `length_append` THEN Auto
. |
. .\
. . 1. E : TaggedEventStruct
. . 2. tr : |E| List
. . 3. Q : ℕ||tr||→Prop
. . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . 5. i : ℕ||tr||
. . 6. Q(i)
. . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . 10. ∀i,j:ℕ||tr||.
. . |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . 11. Causal(E)(tr)
. . 12. No-dup-send(E)(tr)
. . 13. ¬tr = nil
. . 14. L_1 : |E| List
. . 15. L_2 : |E| List
. . 16. tr = (L_1 @ L_2)
. . 17. ∀i:ℕ||tr||.
. . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . 18. ||tr|| = ||L_1||+||L_2||
. . ⊢  ∃L_1,L_2:Trace(E).
. . |  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))  by Assert ||L_1||≤i
. . |
. . |\
. . | .....assertion .....
. . | 1. E : TaggedEventStruct
. . | 2. tr : |E| List
. . | 3. Q : ℕ||tr||→Prop
```

```
. . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . | 5. i : ℕ||tr||
. . | 6. Q(i)
. . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . | 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . | 10. ∀i,j:ℕ||tr||.
. . | |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . | 11. Causal(E)(tr)
. . | 12. No-dup-send(E)(tr)
. . | 13. ¬tr = nil
. . | 14. L_1 : |E| List
. . | 15. L_2 : |E| List
. . | 16. tr = (L_1 @ L_2)
. . | 17. ∀i:ℕ||tr||.
. . | |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . | 18. ||tr|| = ||L_1||+||L_2||
. . | ⊢  ||L_1||≤i  by BackThruSomeHyp THEN InstConcl [i] THEN Auto
. . | |
. . | 1. E : TaggedEventStruct
. . | 2. tr : |E| List
. . | 3. Q : ℕ||tr||→Prop
. . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . | 5. i : ℕ||tr||
. . | 6. Q(i)
. . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . | 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . | 10. ∀i,j:ℕ||tr||.
. . |     (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . | 11. Causal(E)(tr)
. . | 12. No-dup-send(E)(tr)
. . | 13. ¬tr = nil
. . | 14. L_1 : |E| List
. . | 15. L_2 : |E| List
. . | 16. tr = (L_1 @ L_2)
. . | 17. ∀i:ℕ||tr||.
. . |     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . | 18. ||tr|| = ||L_1||+||L_2||
. . | ⊢  tr[i] =msg=(E) tr[i]
. . |   by InstLemma ‘event_msg_eq_equiv‘ [E] THENA Auto THEN UseEquiv
. . |
. . |
. . .\
. . . 1. E : TaggedEventStruct
. . . 2. tr : |E| List
. . . 3. Q : ℕ||tr||→Prop
. . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . 5. i : ℕ||tr||
. . . 6. Q(i)
. . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . 10. ∀i,j:ℕ||tr||.
. . . |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
```

```
. . . 11. Causal(E)(tr)
. . . 12. No-dup-send(E)(tr)
. . . 13. ¬tr = nil
. . . 14. L_1 : |E| List
. . . 15. L_2 : |E| List
. . . 16. tr = (L_1 @ L_2)
. . . 17. ∀i:ℕ||tr||.
. . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . 18. ||tr|| = ||L_1||+||L_2||
. . . 19. ||L_1||≤i
. . . ⊢  ∃L_1,L_2:Trace(E).
. . . |  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))
. . . | by Assert (tr[i] ∈ L_2)
. . . |
. . . |\
. . . | .....assertion .....
. . . | 1. E : TaggedEventStruct
. . . | 2. tr : |E| List
. . . | 3. Q : ℕ||tr||→Prop
. . . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . | 5. i : ℕ||tr||
. . . | 6. Q(i)
. . . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . | 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . | 10. ∀i,j:ℕ||tr||.
. . . | |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . | 11. Causal(E)(tr)
. . . | 12. No-dup-send(E)(tr)
. . . | 13. ¬tr = nil
. . . | 14. L_1 : |E| List
. . . | 15. L_2 : |E| List
. . . | 16. tr = (L_1 @ L_2)
. . . | 17. ∀i:ℕ||tr||.
. . . | |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . | 18. ||tr|| = ||L_1||+||L_2||
. . . | 19. ||L_1||≤i
. . . | ⊢  (tr[i] ∈ L_2)
. . . | | by SubstFor tr 0 THENA Auto'
. . . |
. . . | |    THEN RW0 "select_append_back" 0 THEN Auto
. . . | |
. . . | 1. E : TaggedEventStruct
. . . | 2. tr : |E| List
. . . | 3. Q : ℕ||tr||→Prop
. . . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . | 5. i : ℕ||tr||
. . . | 6. Q(i)
. . . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . | 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . | 10. ∀i,j:ℕ||tr||.
```

```
. . . |     (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . | 11. Causal(E)(tr)
. . . | 12. No-dup-send(E)(tr)
. . . | 13. ¬tr = nil
. . . | 14. L_1 : |E| List
. . . | 15. L_2 : |E| List
. . . | 16. tr = (L_1 @ L_2)
. . . | 17. ∀i:ℕ||tr||.
. . . |     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . | 18. ||tr|| = ||L_1||+||L_2||
. . . | 19. ||L_1||≤i
. . . | ⊢  (L_2[(i-||L_1||)] ∈ L_2)
. . . |   by Unfold `l_member` 0 THEN InstConcl [i-||L_1||
. . . |      ] THEN Auto'
. . . |
. . . .\
. . . . 1. E : TaggedEventStruct
. . . . 2. tr : |E| List
. . . . 3. Q : ℕ||tr||→Prop
. . . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . 5. i : ℕ||tr||
. . . . 6. Q(i)
. . . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . 8. ∀i,j:ℕ||tr||. Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . 10. ∀i,j:ℕ||tr||.
. . . . |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . 11. Causal(E)(tr)
. . . . 12. No-dup-send(E)(tr)
. . . . 13. ¬tr = nil
. . . . 14. L_1 : |E| List
. . . . 15. L_2 : |E| List
. . . . 16. tr = (L_1 @ L_2)
. . . . 17. ∀i:ℕ||tr||.
. . . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . . 18. ||tr|| = ||L_1||+||L_2||
. . . . 19. ||L_1||≤i
. . . . 20. (tr[i] ∈ L_2)
. . . . ⊢  ∃L_1,L_2:Trace(E).
. . . . |  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))
. . . . | by Unfold `str_trace` 0 THEN InstConcl [L_1;L_2] THEN Auto
. . . . |
. . . . |\
. . . . | 1. E : TaggedEventStruct
. . . . | 2. tr : |E| List
. . . . | 3. Q : ℕ||tr||→Prop
. . . . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | 5. i : ℕ||tr||
. . . . | 6. Q(i)
. . . . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | 8. ∀i,j:ℕ||tr||.
. . . . |    Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
```

```
. . . . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | 10. ∀i,j:ℕ||tr||.
. . . . |     (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | 11. Causal(E)(tr)
. . . . | 12. No-dup-send(E)(tr)
. . . . | 13. ¬tr = nil
. . . . | 14. L_1 : |E| List
. . . . | 15. L_2 : |E| List
. . . . | 16. tr = (L_1 @ L_2)
. . . . | 17. ∀i:ℕ||tr||.
. . . . |     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . . | 18. ||tr|| = ||L_1||+||L_2||
. . . . | 19. ||L_1||≤i
. . . . | 20. (tr[i] ∈ L_2)
. . . . | ⊢  ¬L_2 = nil
. . . . |   by AssertBY ¬null(L_2)
. . . . |
. . . . |          (BackThruLemma' `member_null`  THEN Auto)
. . . . |
. . . . |       THEN RW assert_pushdownC (-1) THEN Auto
. . . . |
. . . . |\
. . . . | 1. E : TaggedEventStruct
. . . . | 2. tr : |E| List
. . . . | 3. Q : ℕ||tr||→Prop
. . . . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | 5. i : ℕ||tr||
. . . . | 6. Q(i)
. . . . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | 8. ∀i,j:ℕ||tr||.
. . . . | |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | 10. ∀i,j:ℕ||tr||.
. . . . | |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | 11. Causal(E)(tr)
. . . . | 12. No-dup-send(E)(tr)
. . . . | 13. ¬tr = nil
. . . . | 14. L_1 : |E| List
. . . . | 15. L_2 : |E| List
. . . . | 16. tr = (L_1 @ L_2)
. . . . | 17. ∀i:ℕ||tr||.
. . . . | |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . . | 18. ||tr|| = ||L_1||+||L_2||
. . . . | 19. ||L_1||≤i
. . . . | 20. (tr[i] ∈ L_2)
. . . . | ⊢  (∀x∈L_1.(∀y∈L_2.¬(x =msg=(E) y)))
. . . . | | by ...
. . . . | THEN SubstFor x 0 THEN SubstFor y
. . . . | |     0 THEN D 0 THENA Auto
. . . . | |
. . . . | 1. E : TaggedEventStruct
. . . . | 2. tr : |E| List
. . . . | 3. Q : ℕ||tr||→Prop
. . . . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | 5. i : ℕ||tr||
```

```
. . . . | 6. Q(i)
. . . . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | 8. ∀i,j:ℕ||tr||.
. . . . | |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | 10. ∀i,j:ℕ||tr||.
. . . . | |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | 11. Causal(E)(tr)
. . . . | 12. No-dup-send(E)(tr)
. . . . | 13. ¬tr = nil
. . . . | 14. L_1 : |E| List
. . . . | 15. L_2 : |E| List
. . . . | 16. tr = (L_1 @ L_2)
. . . . | 17. ∀i:ℕ||tr||.
. . . . | |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . . | 18. ||tr|| = ||L_1||+||L_2||
. . . . | 19. ||L_1||≤i
. . . . | 20. (tr[i] ∈ L_2)
. . . . | 21. x : |E|
. . . . | 22. i2 : ℕ
. . . . | 23. i2<||L_1||
. . . . | 24. x = L_1[i2]
. . . . | 25. y : |E|
. . . . | 26. i1 : ℕ
. . . . | 27. i1<||L_2||
. . . . | 28. y = L_2[i1]
. . . . | 29. L_1[i2] =msg=(E) L_2[i1]
. . . . | ⊢  False
. . . . | | by Subst L_2[i1] = tr[(i1+||L_1||)] (-1) THENA Auto'
. . . . | |
. . . . | |\
. . . . | | .....equality .....
. . . . | | 1. E : TaggedEventStruct
. . . . | | 2. tr : |E| List
. . . . | | 3. Q : ℕ||tr||→Prop
. . . . | | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | | 5. i : ℕ||tr||
. . . . | | 6. Q(i)
. . . . | | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | | 8. ∀i,j:ℕ||tr||.
. . . . | |    Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | | 10. ∀i,j:ℕ||tr||.
. . . . | |     (tr[i] =msg=(E) tr[j]) ⇒
. . . . | |     tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | | 11. Causal(E)(tr)
. . . . | | 12. No-dup-send(E)(tr)
. . . . | | 13. ¬tr = nil
. . . . | | 14. L_1 : |E| List
. . . . | | 15. L_2 : |E| List
. . . . | | 16. tr = (L_1 @ L_2)
. . . . | | 17. ∀i:ℕ||tr||.
. . . . | |     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . | |     ⟺
. . . . | |     ||L_1||≤i
```

```
. . . . | | 18. ||tr|| = ||L_1||+||L_2||
. . . . | | 19. ||L_1||≤i
. . . . | | 20. (tr[i] ∈ L_2)
. . . . | | 21. x : |E|
. . . . | | 22. i2 : ℕ
. . . . | | 23. i2<||L_1||
. . . . | | 24. x = L_1[i2]
. . . . | | 25. y : |E|
. . . . | | 26. i1 : ℕ
. . . . | | 27. i1<||L_2||
. . . . | | 28. y = L_2[i1]
. . . . | | 29. L_1[i2] =msg=(E) L_2[i1]
. . . . | | ⊢  L_2[i1] = tr[(i1+||L_1||)]
. . . . | |   by SubstFor tr
. . . . | |      ...
. . . . | |
. . . . | .\
. . . . | . 1. E : TaggedEventStruct
. . . . | . 2. tr : |E| List
. . . . | . 3. Q : ℕ||tr||→Prop
. . . . | . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | . 5. i : ℕ||tr||
. . . . | . 6. Q(i)
. . . . | . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | . 8. ∀i,j:ℕ||tr||.
. . . . | . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | . 10. ∀i,j:ℕ||tr||.
. . . . | . |   (tr[i] =msg=(E) tr[j]) ⇒
. . . . | . |   tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . 11. Causal(E)(tr)
. . . . | . 12. No-dup-send(E)(tr)
. . . . | . 13. ¬tr = nil
. . . . | . 14. L_1 : |E| List
. . . . | . 15. L_2 : |E| List
. . . . | . 16. tr = (L_1 @ L_2)
. . . . | . 17. ∀i:ℕ||tr||.
. . . . | . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . | . |   ⟺
. . . . | . |   ||L_1||≤i
. . . . | . 18. ||tr|| = ||L_1||+||L_2||
. . . . | . 19. ||L_1||≤i
. . . . | . 20. (tr[i] ∈ L_2)
. . . . | . 21. x : |E|
. . . . | . 22. i2 : ℕ
. . . . | . 23. i2<||L_1||
. . . . | . 24. x = L_1[i2]
. . . . | . 25. y : |E|
. . . . | . 26. i1 : ℕ
. . . . | . 27. i1<||L_2||
. . . . | . 28. y = L_2[i1]
. . . . | . 29. L_1[i2] =msg=(E) tr[(i1+||L_1||)]
. . . . | . ⊢  False  by Subst L_1[i2] = tr[i2] (-1) THENA Auto'
. . . . | . |
. . . . | . |\
```

```
. . . . | . | .....equality .....
. . . . | . | 1. E : TaggedEventStruct
. . . . | . | 2. tr : |E| List
. . . . | . | 3. Q : ℕ||tr||→Prop
. . . . | . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | . | 5. i : ℕ||tr||
. . . . | . | 6. Q(i)
. . . . | . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | . | 8. ∀i,j:ℕ||tr||.
. . . . | . |    Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | . | 10. ∀i,j:ℕ||tr||.
. . . . | . |     (tr[i] =msg=(E) tr[j]) ⇒
. . . . | . |     tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . | 11. Causal(E)(tr)
. . . . | . | 12. No-dup-send(E)(tr)
. . . . | . | 13. ¬tr = nil
. . . . | . | 14. L_1 : |E| List
. . . . | . | 15. L_2 : |E| List
. . . . | . | 16. tr = (L_1 @ L_2)
. . . . | . | 17. ∀i:ℕ||tr||.
. . . . | . |     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . | . |     ⟺
. . . . | . |     ||L_1||≤i
. . . . | . | 18. ||tr|| = ||L_1||+||L_2||
. . . . | . | 19. ||L_1||≤i
. . . . | . | 20. (tr[i] ∈ L_2)
. . . . | . | 21. x : |E|
. . . . | . | 22. i2 : ℕ
. . . . | . | 23. i2<||L_1||
. . . . | . | 24. x = L_1[i2]
. . . . | . | 25. y : |E|
. . . . | . | 26. i1 : ℕ
. . . . | . | 27. i1<||L_2||
. . . . | . | 28. y = L_2[i1]
. . . . | . | 29. L_1[i2] =msg=(E) tr[(i1+||L_1||)]
. . . . | . | ⊢  L_1[i2] = tr[i2]
. . . . | . |   by SubstFor tr
. . . . | . |      ...
. . . . | . |
. . . . | . .\
. . . . | . . 1. E : TaggedEventStruct
. . . . | . . 2. tr : |E| List
. . . . | . . 3. Q : ℕ||tr||→Prop
. . . . | . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | . . 5. i : ℕ||tr||
. . . . | . . 6. Q(i)
. . . . | . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | . . 8. ∀i,j:ℕ||tr||.
. . . . | . . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | . . 10. ∀i,j:ℕ||tr||.
. . . . | . . |   (tr[i] =msg=(E) tr[j]) ⇒
. . . . | . . |   tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . . 11. Causal(E)(tr)
```

```
. . . . | . . 12. No-dup-send(E)(tr)
. . . . | . . 13. ¬tr = nil
. . . . | . . 14. L_1 : |E| List
. . . . | . . 15. L_2 : |E| List
. . . . | . . 16. tr = (L_1 @ L_2)
. . . . | . . 17. ∀i:ℕ||tr||.
. . . . | . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . | . . |   ⟺
. . . . | . . |   ||L_1||≤i
. . . . | . . 18. ||tr|| = ||L_1||+||L_2||
. . . . | . . 19. ||L_1||≤i
. . . . | . . 20. (tr[i] ∈ L_2)
. . . . | . . 21. x : |E|
. . . . | . . 22. i2 : ℕ
. . . . | . . 23. i2<||L_1||
. . . . | . . 24. x = L_1[i2]
. . . . | . . 25. y : |E|
. . . . | . . 26. i1 : ℕ
. . . . | . . 27. i1<||L_2||
. . . . | . . 28. y = L_2[i1]
. . . . | . . 29. tr[i2] =msg=(E) tr[(i1+||L_1||)]
. . . . | . . ⊢  False
. . . . | . . | by InstHyp [i1+||L_1||] 17 THENA Complete Auto'
. . . . | . .
. . . . | . . |    ...
. . . . | . . THEN InstHyp [i2] 17 THENA Complete Auto'
. . . . | . .
. . . . | . . |    THEN D (-1) THEN D (-2) THEN Auto'
. . . . | . .
. . . . | . . |    THEN InstConcl [k] THEN Auto
. . . . | . . |
. . . . | . . 1. E : TaggedEventStruct
. . . . | . . 2. tr : |E| List
. . . . | . . 3. Q : ℕ||tr||→Prop
. . . . | . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | . . 5. i : ℕ||tr||
. . . . | . . 6. Q(i)
. . . . | . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | . . 8. ∀i,j:ℕ||tr||.
. . . . | . . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | . . 10. ∀i,j:ℕ||tr||.
. . . . | . . |   (tr[i] =msg=(E) tr[j]) ⇒
. . . . | . . |   tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . . 11. Causal(E)(tr)
. . . . | . . 12. No-dup-send(E)(tr)
. . . . | . . 13. ¬tr = nil
. . . . | . . 14. L_1 : |E| List
. . . . | . . 15. L_2 : |E| List
. . . . | . . 16. tr = (L_1 @ L_2)
. . . . | . . 17. ∀i:ℕ||tr||.
. . . . | . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . | . . |   ⟺
. . . . | . . |   ||L_1||≤i
. . . . | . . 18. ||tr|| = ||L_1||+||L_2||
```

```
. . . . | . . 19. ||L_1||≤i
. . . . | . . 20. (tr[i] ∈ L_2)
. . . . | . . 21. x : |E|
. . . . | . . 22. i2 : ℕ
. . . . | . . 23. i2<||L_1||
. . . . | . . 24. x = L_1[i2]
. . . . | . . 25. y : |E|
. . . . | . . 26. i1 : ℕ
. . . . | . . 27. i1<||L_2||
. . . . | . . 28. y = L_2[i1]
. . . . | . . 29. tr[i2] =msg=(E) tr[(i1+||L_1||)]
. . . . | . . 30. k : ℕ||tr||
. . . . | . . 31. Q(k)
. . . . | . . 32. tr[k] =msg=(E) tr[(i1+||L_1||)]
. . . . | . . 33. (∃k:ℕ||tr||.
. . . . | . .  |    Q(k) & (tr[k] =msg=(E) tr[i2])) ⇐ (||L_1||≤i2)
. . . . | . . ⊢  tr[k] =msg=(E) tr[i2]
. . . . | . . | by InstLemma ‘event_msg_eq_equiv‘ [E] THENA Auto
. . . . | . . |
. . . . | . . 1. E : TaggedEventStruct
. . . . | . . 2. tr : |E| List
. . . . | . . 3. Q : ℕ||tr||→Prop
. . . . | . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . | . . 5. i : ℕ||tr||
. . . . | . . 6. Q(i)
. . . . | . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . | . . 8. ∀i,j:ℕ||tr||.
. . . . | . .    Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . | . . 10. ∀i,j:ℕ||tr||.
. . . . | . .     (tr[i] =msg=(E) tr[j]) ⇒
. . . . | . .     tag(E)(tr[i]) = tag(E)(tr[j])
. . . . | . . 11. Causal(E)(tr)
. . . . | . . 12. No-dup-send(E)(tr)
. . . . | . . 13. ¬tr = nil
. . . . | . . 14. L_1 : |E| List
. . . . | . . 15. L_2 : |E| List
. . . . | . . 16. tr = (L_1 @ L_2)
. . . . | . . 17. ∀i:ℕ||tr||.
. . . . | . .     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . | . .     ⟺
. . . . | . .     ||L_1||≤i
. . . . | . . 18. ||tr|| = ||L_1||+||L_2||
. . . . | . . 19. ||L_1||≤i
. . . . | . . 20. (tr[i] ∈ L_2)
. . . . | . . 21. x : |E|
. . . . | . . 22. i2 : ℕ
. . . . | . . 23. i2<||L_1||
. . . . | . . 24. x = L_1[i2]
. . . . | . . 25. y : |E|
. . . . | . . 26. i1 : ℕ
. . . . | . . 27. i1<||L_2||
. . . . | . . 28. y = L_2[i1]
. . . . | . . 29. tr[i2] =msg=(E) tr[(i1+||L_1||)]
. . . . | . . 30. k : ℕ||tr||
```

```
. . . . | . . 31. Q(k)
. . . . | . . 32. tr[k] =msg=(E) tr[(i1+||L_1||)]
. . . . | . . 33. (∃k:ℕ||tr||.
. . . . | . .      Q(k) & (tr[k] =msg=(E) tr[i2])) ⇐ (||L_1||≤i2)
. . . . | . . 34. EquivRel(|E|)(_1 =msg=(E) _2)
. . . . | . . ⊢  tr[k] =msg=(E) tr[i2]
. . . . | . .   by UseTrans tr[(i1+||L_1||)]
. . . . | . .       THEN Auto' THEN UseSym THEN Auto'
. . . . |
. . . . .\
. . . . . 1. E : TaggedEventStruct
. . . . . 2. tr : |E| List
. . . . . 3. Q : ℕ||tr||→Prop
. . . . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . . 5. i : ℕ||tr||
. . . . . 6. Q(i)
. . . . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . . 8. ∀i,j:ℕ||tr||.
. . . . . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . . 10. ∀i,j:ℕ||tr||.
. . . . . |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 11. Causal(E)(tr)
. . . . . 12. No-dup-send(E)(tr)
. . . . . 13. ¬tr = nil
. . . . . 14. L_1 : |E| List
. . . . . 15. L_2 : |E| List
. . . . . 16. tr = (L_1 @ L_2)
. . . . . 17. ∀i:ℕ||tr||.
. . . . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . . . 18. ||tr|| = ||L_1||+||L_2||
. . . . . 19. ||L_1||≤i
. . . . . 20. (tr[i] ∈ L_2)
. . . . . ⊢  ∃m:Label. (∀x∈L_2.tag(E)(x) = m)
. . . . . | by InstConcl [tag(E)(tr[i])] THENA Auto
. . . . .
. . . . . |    THEN Unfold `l_all` 0 THEN Auto
. . . . .
. . . . . |    THEN D (-1) THEN ExRepD THEN SubstFor x 0
. . . . . |
. . . . . |
. . . . . 1. E : TaggedEventStruct
. . . . . 2. tr : |E| List
. . . . . 3. Q : ℕ||tr||→Prop
. . . . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . . 5. i : ℕ||tr||
. . . . . 6. Q(i)
. . . . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . . 8. ∀i,j:ℕ||tr||.
. . . . . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . . 10. ∀i,j:ℕ||tr||.
. . . . . |   (tr[i] =msg=(E) tr[j]) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 11. Causal(E)(tr)
. . . . . 12. No-dup-send(E)(tr)
```

```
. . . . . 13. ¬tr = nil
. . . . . 14. L_1 : |E| List
. . . . . 15. L_2 : |E| List
. . . . . 16. tr = (L_1 @ L_2)
. . . . . 17. ∀i:ℕ||tr||.
. . . . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i])) ⟺ ||L_1||≤i
. . . . . 18. ||tr|| = ||L_1||+||L_2||
. . . . . 19. ||L_1||≤i
. . . . . 20. (tr[i] ∈ L_2)
. . . . . 21. x : |E|
. . . . . 22. i1 : ℕ
. . . . . 23. i1<||L_2||
. . . . . 24. x = L_2[i1]
. . . . . ⊢  tag(E)(L_2[i1]) = tag(E)(tr[i])
. . . . . | by Subst L_2[i1] = tr[(i1+||L_1||)] 0 THENA Auto
. . . . . |
. . . . . |\
. . . . . | .....equality .....
. . . . . | 1. E : TaggedEventStruct
. . . . . | 2. tr : |E| List
. . . . . | 3. Q : ℕ||tr||→Prop
. . . . . | 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . . | 5. i : ℕ||tr||
. . . . . | 6. Q(i)
. . . . . | 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . . | 8. ∀i,j:ℕ||tr||.
. . . . . |    Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . | 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . . | 10. ∀i,j:ℕ||tr||.
. . . . . |     (tr[i] =msg=(E) tr[j]) ⇒
. . . . . |     tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . | 11. Causal(E)(tr)
. . . . . | 12. No-dup-send(E)(tr)
. . . . . | 13. ¬tr = nil
. . . . . | 14. L_1 : |E| List
. . . . . | 15. L_2 : |E| List
. . . . . | 16. tr = (L_1 @ L_2)
. . . . . | 17. ∀i:ℕ||tr||.
. . . . . |     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . . |     ⟺
. . . . . |     ||L_1||≤i
. . . . . | 18. ||tr|| = ||L_1||+||L_2||
. . . . . | 19. ||L_1||≤i
. . . . . | 20. (tr[i] ∈ L_2)
. . . . . | 21. x : |E|
. . . . . | 22. i1 : ℕ
. . . . . | 23. i1<||L_2||
. . . . . | 24. x = L_2[i1]
. . . . . | ⊢  L_2[i1] = tr[(i1+||L_1||)]
. . . . . |   by SubstFor tr
. . . . . |      ...
. . . . . |
. . . . . .\
. . . . . . 1. E : TaggedEventStruct
. . . . . . 2. tr : |E| List
```

```
. . . . . 3. Q : ℕ||tr||→Prop
. . . . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . . 5. i : ℕ||tr||
. . . . . 6. Q(i)
. . . . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . . 8. ∀i,j:ℕ||tr||.
. . . . . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . . 10. ∀i,j:ℕ||tr||.
. . . . . |   (tr[i] =msg=(E) tr[j]) ⇒
. . . . . |   tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 11. Causal(E)(tr)
. . . . . 12. No-dup-send(E)(tr)
. . . . . 13. ¬tr = nil
. . . . . 14. L_1 : |E| List
. . . . . 15. L_2 : |E| List
. . . . . 16. tr = (L_1 @ L_2)
. . . . . 17. ∀i:ℕ||tr||.
. . . . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . . |   ⟺
. . . . . |   ||L_1||≤i
. . . . . 18. ||tr|| = ||L_1||+||L_2||
. . . . . 19. ||L_1||≤i
. . . . . 20. (tr[i] ∈ L_2)
. . . . . 21. x : |E|
. . . . . 22. i1 : ℕ
. . . . . 23. i1<||L_2||
. . . . . 24. x = L_2[i1]
. . . . . ⊢  tag(E)(tr[(i1+||L_1||)]) = tag(E)(tr[i])
. . . . . | by InstHyp [i1+||L_1||] (-8) THENA Auto'
. . . . . |
. . . . . 1. E : TaggedEventStruct
. . . . . 2. tr : |E| List
. . . . . 3. Q : ℕ||tr||→Prop
. . . . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . . 5. i : ℕ||tr||
. . . . . 6. Q(i)
. . . . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . . 8. ∀i,j:ℕ||tr||.
. . . . . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . . 10. ∀i,j:ℕ||tr||.
. . . . . |   (tr[i] =msg=(E) tr[j]) ⇒
. . . . . |   tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 11. Causal(E)(tr)
. . . . . 12. No-dup-send(E)(tr)
. . . . . 13. ¬tr = nil
. . . . . 14. L_1 : |E| List
. . . . . 15. L_2 : |E| List
. . . . . 16. tr = (L_1 @ L_2)
. . . . . 17. ∀i:ℕ||tr||.
. . . . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . . |   ⟺
. . . . . |   ||L_1||≤i
. . . . . 18. ||tr|| = ||L_1||+||L_2||
```

```
. . . . . 19. ||L_1||≤i
. . . . . 20. (tr[i] ∈ L_2)
. . . . . 21. x : |E|
. . . . . 22. i1 : ℕ
. . . . . 23. i1<||L_2||
. . . . . 24. x = L_2[i1]
. . . . . 25. (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[(i1+||L_1||)]))
. . . . . |   ⟺
. . . . . |   ||L_1||≤i1+||L_1||
. . . . . ⊢  tag(E)(tr[(i1+||L_1||)]) = tag(E)(tr[i])
. . . . . | by D (-1) THEN D (-1) THENA Auto' THEN ExRepD
. . . . . |
. . . . . 1. E : TaggedEventStruct
. . . . . 2. tr : |E| List
. . . . . 3. Q : ℕ||tr||→Prop
. . . . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . . 5. i : ℕ||tr||
. . . . . 6. Q(i)
. . . . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . . 8. ∀i,j:ℕ||tr||.
. . . . . |  Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . . 10. ∀i,j:ℕ||tr||.
. . . . . |   (tr[i] =msg=(E) tr[j]) ⇒
. . . . . |   tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 11. Causal(E)(tr)
. . . . . 12. No-dup-send(E)(tr)
. . . . . 13. ¬tr = nil
. . . . . 14. L_1 : |E| List
. . . . . 15. L_2 : |E| List
. . . . . 16. tr = (L_1 @ L_2)
. . . . . 17. ∀i:ℕ||tr||.
. . . . . |   (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . . |   ⟺
. . . . . |   ||L_1||≤i
. . . . . 18. ||tr|| = ||L_1||+||L_2||
. . . . . 19. ||L_1||≤i
. . . . . 20. (tr[i] ∈ L_2)
. . . . . 21. x : |E|
. . . . . 22. i1 : ℕ
. . . . . 23. i1<||L_2||
. . . . . 24. x = L_2[i1]
. . . . . 25. (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[(i1+||L_1||)]))
. . . . . |   ⇒
. . . . . |   ||L_1||≤i1+||L_1||
. . . . . 26. k : ℕ||tr||
. . . . . 27. Q(k)
. . . . . 28. tr[k] =msg=(E) tr[(i1+||L_1||)]
. . . . . ⊢  tag(E)(tr[(i1+||L_1||)]) = tag(E)(tr[i])
. . . . . | by Subst tag(E)(tr[(i1+||L_1||)]) = tag(E)(tr[k])
. . . . . |     0 THEN Auto THEN EasyHyp
. . . . . |
. . . . . 1. E : TaggedEventStruct
. . . . . 2. tr : |E| List
. . . . . 3. Q : ℕ||tr||→Prop
```

```
. . . . . 4. ∀i:ℕ||tr||. Dec(Q(i))
. . . . . 5. i : ℕ||tr||
. . . . . 6. Q(i)
. . . . . 7. ∀i:ℕ||tr||. Q(i) ⇒ is-send(E)(tr[i])
. . . . . 8. ∀i,j:ℕ||tr||.
. . . . .    Q(i) ⇒ Q(j) ⇒ tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 9. ∀i,j:ℕ||tr||. Q(i) ⇒ i≤j ⇒ C(Q)(j)
. . . . . 10. ∀i,j:ℕ||tr||.
. . . . .     (tr[i] =msg=(E) tr[j]) ⇒
. . . . .     tag(E)(tr[i]) = tag(E)(tr[j])
. . . . . 11. Causal(E)(tr)
. . . . . 12. No-dup-send(E)(tr)
. . . . . 13. ¬tr = nil
. . . . . 14. L_1 : |E| List
. . . . . 15. L_2 : |E| List
. . . . . 16. tr = (L_1 @ L_2)
. . . . . 17. ∀i:ℕ||tr||.
. . . . .     (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[i]))
. . . . .     ⟺
. . . . .     ||L_1||≤i
. . . . . 18. ||tr|| = ||L_1||+||L_2||
. . . . . 19. ||L_1||≤i
. . . . . 20. (tr[i] ∈ L_2)
. . . . . 21. x : |E|
. . . . . 22. i1 : ℕ
. . . . . 23. i1<||L_2||

. . . . . 24. x = L_2[i1]
. . . . . 25. (∃k:ℕ||tr||. Q(k) & (tr[k] =msg=(E) tr[(i1+||L_1||)]))
. . . . .     ⇒
. . . . .     ||L_1||≤i1+||L_1||
. . . . . 26. k : ℕ||tr||
. . . . . 27. Q(k)
. . . . . 28. tr[k] =msg=(E) tr[(i1+||L_1||)]
. . . . . ⊢  tag(E)(tr[(i1+||L_1||)]) = tag(E)(tr[k])
. . . . .   by Symmetry THEN BackThruSomeHyp THEN Auto'
END
```