```
switchable_induced  THM

⊢  ∀E:EventStruct, A:Type, f:(A→|E|), P:((|E| List)→Prop).
|  switchable(E)(P) ⇒ switchable(induced_event_str(E;A;f))(P o f)
| by Auto THEN All (Unfolds ``b_switchable compose_map``)

|    THEN All Reduce

|    THEN All (Unfolds ``preserved_by preserved_by2 tr_refines``)

|    THEN All Reduce
THEN Repeat (ParallelAnd (-1))
|
|\
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y:|E| List. P(x) ⇒ (x safetyR(E) y) ⇒ P(y)
| ⊢  ∀x,y:A List.
|    P(map(f;x)) ⇒
|    (x safetyR(induced_event_str(E;A;f)) y) ⇒ P(map(f;y))
|   by Auto THEN Using [`x',map(f;x)] (BackThruSomeHyp) THEN Auto
|
|      THEN All (Unfold `R_safety`) THEN All Reduce THEN Easy
|
|\
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y:|E| List. P(x) ⇒ (x memorylessR(E) y) ⇒ P(y)
| ⊢  ∀x,y:A List.
| |  P(map(f;x)) ⇒
| |  (x memorylessR(induced_event_str(E;A;f)) y) ⇒ P(map(f;y))
| | by Auto THEN Using [`x',map(f;x)] (BackThruSomeHyp) THEN Auto
|
| |    THEN All (Unfold `R_memoryless`) THEN All Reduce
| |
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y:|E| List.
| |  P(x) ⇒ (∃a:|E|. y = filter(λb.¬₂(b =msg=(E) a);x)) ⇒ P(y)
| 6. x : A List
| 7. y : A List
| 8. P(map(f;x))
| 9. ∃a:A. y = filter(λb.¬₂(b =msg=(induced_event_str(E;A;f)) a);x)
| ⊢  ∃a:|E|. map(f;y) = filter(λb.¬₂(b =msg=(E) a);map(f;x))
| | by ExRepD THEN InstConcl [f(a)
| |    ] THEN Auto THEN HypSubst (-1) 0 THENA Auto
|
| |    THEN RWO "filter_map" 0 THENA Auto
| |
```

```
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y:|E| List.
|    P(x) ⇒ (∃a:|E|. y = filter(λb.¬₂(b =msg=(E) a);x)) ⇒ P(y)
| 6. x : A List
| 7. y : A List
| 8. P(map(f;x))
| 9. a : A
| 10. y = filter(λb.¬₂(b =msg=(induced_event_str(E;A;f)) a);x)
| ⊢  map(f;filter(λb.¬₂(b =msg=(induced_event_str(E;A;f)) a);x))
|    =
|    map(f;filter((λb.¬₂(b =msg=(E) (f(a)))) o f;x))
|   by Unfolds ``compose induced_event_str`` 0 THEN Reduce 0
|
|      THEN All (Unfold `event_msg_eq`) THEN All Reduce THEN Auto
|
|\
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y,z:|E| List. P(x) ⇒ P(y) ⇒ composableR(E)(x,y,z) ⇒ P(z)
| ⊢  ∀x,y,z:A List.
| |  P(map(f;x)) ⇒
| |  P(map(f;y)) ⇒
| |  composableR(induced_event_str(E;A;f))(x,y,z) ⇒ P(map(f;z))
| | by Auto THEN Using [`x',map(f;x); `y',map(f;y)
| |    ] (BackThruSomeHyp) THEN Auto
| |
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y,z:|E| List. P(x) ⇒ P(y) ⇒ composableR(E)(x,y,z) ⇒ P(z)
| 6. x : A List
| 7. y : A List
| 8. z : A List
| 9. P(map(f;x))
| 10. P(map(f;y))
| 11. composableR(induced_event_str(E;A;f))(x,y,z)
| ⊢  composableR(E)(map(f;x),map(f;y),map(f;z))
| | by All (Unfold `R_composable`) THEN All Reduce
| |
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 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 : A List
| 7. y : A List
| 8. z : A List
```

```
| 9. P(map(f;x))
| 10. P(map(f;y))
| 11. (∀x∈x.(∀y∈y.¬(x =msg=(induced_event_str(E;A;f)) y)))
| |    & z = (x @ y)
| ⊢  (∀x∈map(f;x).(∀y∈map(f;y).¬(x =msg=(E) y)))
| |  & map(f;z) = (map(f;x) @ map(f;y))
| | by RWW "l_all_map" 0 THENA Auto THEN ParallelOp (-1)
|
| |    ...
| THEN RepeatFor 2 (ParallelOp (-1)) THEN All (Fold `l_all`)
| |
| THEN RWO "l_all_map" 0 THENA Auto
| |
| |
| |
| |
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 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 : A List
| 7. y : A List
| 8. A List
| 9. P(map(f;x))
| 10. P(map(f;y))
| 11. (∀x@0∈x.(∀y@0∈y.¬(x@0 =msg=(induced_event_str(E;A;f)) y@0)))
| 12. x@0 : A
| 13. (x@0 ∈ x) ⇒ (∀y@0∈y.¬(x@0 =msg=(induced_event_str(E;A;f)) y@0))
| ⊢  (x@0 ∈ x) ⇒ (∀y@0∈y.¬((f(x@0)) =msg=(E) (f(y@0))))
|   by ...
| THEN Unfold `event_msg_eq` 0 THEN Reduce 0 THEN Trivial
|
|\
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y:|E| List. P(x) ⇒ (x send-enabledR(E) y) ⇒ P(y)
| ⊢  ∀x,y:A List.
|    P(map(f;x)) ⇒
|    (x send-enabledR(induced_event_str(E;A;f)) y) ⇒ P(map(f;y))
|   by Auto THEN Using [`x',map(f;x); `y',map(f;y)
|      ] (BackThruSomeHyp) THEN Auto
|
|      THEN All (\h. Unfold `R_send_enabled` h THEN Reduce h)
|
|      THEN Unfold `induced_event_str` (-1) THEN Reduce (-1)
|
|      THEN ExRepD THEN InstConcl [f(x@0)] THEN Auto
|
|      THEN HypSubst (-1) 0 THENA Auto
|
```

```
|      THEN RW MapAppendC 0 THEN Reduce 0 THEN Auto
|
|\
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y:|E| List. P(x) ⇒ (x asyncR(E) y) ⇒ P(y)
| ⊢  ∀x,y:A List.
|    P(map(f;x)) ⇒
|    (x asyncR(induced_event_str(E;A;f)) y) ⇒ P(map(f;y))
|   by Auto THEN Using [‘x’,map(f;x); ‘y’,map(f;y)
|      ] (BackThruSomeHyp) THEN Auto
|
|      ...
| THEN  BackThruLemma ‘swap_adjacent_map‘ THEN Auto
|
|\
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀x,y:|E| List. P(x) ⇒ (x delayableR(E) y) ⇒ P(y)
| ⊢  ∀x,y:A List.
|    P(map(f;x)) ⇒
|    (x delayableR(induced_event_str(E;A;f)) y) ⇒ P(map(f;y))
|   by Auto THEN Using [‘x’,map(f;x); ‘y’,map(f;y)
|      ] (BackThruSomeHyp) THEN Auto
|
|      ...
| THEN BackThruLemma ‘swap_adjacent_map‘ THEN Auto
|
|      THEN All (Unfold ‘event_msg_eq‘) THEN All Reduce THEN Trivial
|
|
|
|\
| 1. E : EventStruct
| 2. A : Type
| 3. f : A→|E|
| 4. P : (|E| List)→Prop
| 5. ∀tr:|E| List. P(tr) ⇒ Causal(E)(tr)
| ⊢  ∀tr:A List. P(map(f;tr)) ⇒ Causal(induced_event_str(E;A;f))(tr)
|   by ...
| THEN Auto  THEN InstHyp [map(f;tr);i] (-4) THEN Auto

|
|      THEN Try (RWO "map_length" 0 THEN Auto)
|
|      THEN All (\h. Unfold ‘event_msg_eq‘ h THEN Reduce h)
|
|      THEN ParallelOp (-1)
|
|      THEN Try (RWO "map_select" (-1) THEN Auto)
|
.\
```

```
. 1. E : EventStruct
. 2. A : Type
. 3. f : A→|E|
. 4. P : (|E| List)→Prop
. 5. ∀tr:|E| List. P(tr) ⇒ No-dup-deliver(E)(tr)
. ⊢  ∀tr:A List.
.     P(map(f;tr)) ⇒ No-dup-deliver(induced_event_str(E;A;f))(tr)
.   by ...
. THEN All(\h. Unfold ‘event_msg_eq‘ h THEN Reduce h)
.
.      THEN Auto THEN AllHyps (InstHyp [map(f;tr);i;j]) THEN Auto
.
.      THEN Try (RWO "map_select" 0 THEN Auto)
.
.      THEN Try (RWO "map_length" 0 THEN Auto)
END
```