```
maction_lemma  THM

⊢  ∀S:ActionSet(Alph), s:S.car, L1,L2,L:Alph*.
|  (S:L1←s) = (S:L2←s) ⇒ (S:L @ L1←s) = (S:L @ L2←s)  by UnivCD
|
1. Alph : Type
2. S : ActionSet(Alph)
3. s : S.car
4. L1 : Alph*
5. L2 : Alph*
6. L : Alph*
7. (S:L1←s) = (S:L2←s)
⊢  (S:L @ L1←s) = (S:L @ L2←s)  by ListInd 6
|
|\
| ⊢  (S:nil @ L1←s) = (S:nil @ L2←s)  by AbReduce 0
|
.\
. 8. u : Alph
. 9. v : Alph*
. 10. (S:v @ L1←s) = (S:v @ L2←s)
. ⊢  (S:(u.v) @ L1←s) = (S:(u.v) @ L2←s)  by RecCaseSplit `maction`
. |
. |\
. | 11. ((u.v) @ L1) = nil
. | ⊢  s = (S:u.(v @ L2)←s)  by AbReduce 11
. | |
. | 11. u.(v @ L1) = nil
. | ⊢
. |   by Inst Thm* ∀h:T, t:T*. ¬h.t = nil    {cons_neq_nil}
. |      [Alph;u;v @ L1]
. |
. .\
. . 11. ¬((u.v) @ L1) = nil
. . ⊢  S.act(u,(S:v @ L1←s)) = (S:u.(v @ L2)←s)
. . | by RW (LowerC (HypC 10)) 0
. . |
. . ⊢  S.act(u,(S:v @ L2←s)) = (S:u.(v @ L2)←s)
. . | by RW (AddrC [3] (RecUnfoldC `maction`)) 0
. . |
. . ⊢  S.act(u,(S:v @ L2←s))
. . |  =
. . |  if null(u.(v @ L2))→ s
. . |  else S.act(hd((u.(v @ L2))),(S:tl((u.(v @ L2)))←s)) fi
. . | by SplitOnConclITE
. . |
. . |\
. . | .....truecase .....
. . | 12. u.(v @ L2) = nil
. . | ⊢  S.act(u,(S:v @ L2←s)) = s
. . |   by Inst Thm* ∀h:T, t:T*. ¬h.t = nil    {cons_neq_nil}
. . |      [Alph;u;v @ L2]
. . |
. . .\
. . . .....falsecase .....
```

```
. . . 12. ¬u.(v @ L2) = nil
. . . ⊢  S.act(u,(S:v @ L2←s))
. . .    =
. . .    S.act(hd((u.(v @ L2))),(S:tl((u.(v @ L2)))←s))
. . .   by AbReduce 0
END
```