```
lang_power_functionality  THM

⊢  ∀L,L':LangOver(Alph), n,n':ℕ. L = L' ⇒ n = n' ⇒ (L↑n) = (L'↑n')
| by UnivCD
|
1. Alph : Type
2. L : LangOver(Alph)
3. L' : LangOver(Alph)
4. n : ℕ
5. n' : ℕ
6. L = L'
7. n = n'
⊢  (L↑n) = (L'↑n')  by RWH (RevHypC 7) 0
|
⊢  (L↑n) = (L'↑n)  by Thin 7
|
6. L = L'
⊢   by NatInd 4
|
|\
| .....basecase .....
| 4. ℕ
| 5. L = L'
| ⊢  (L↑0) = (L'↑0)  by RecCaseSplit `lang_power`
| |
| |\
| | .....truecase .....
| | 6. 0 = 0
| | ⊢  Δ⁰ = (L'↑0)  by RecCaseSplit `lang_power`
| | |
| | .....truecase .....
| | 7. 0 = 0
| | ⊢  Δ⁰ = Δ⁰  by Unfold `lang_eq` 0
| | |
| | ⊢  ∀l:Alph*. Δ⁰(l) ⟺ Δ⁰(l)  by Unfold `lang_sing` 0 THEN UnivCD
| | |
| | 8. l : Alph*
| | ⊢  (λl.null(l))(l) ⟺ (λl.null(l))(l)  by Reduce 0
| |
| .\
| . 6. ¬0 = 0
| . ⊢  ((L↑-1)⊗L) = (L'↑0)  by Auto
|
.\
. .....upcase .....
. 4. ℕ
. 5. L = L'
. 6. n : ℤ
. 7. 0<n
. 8. (L↑n-1) = (L'↑n-1)
. ⊢   by RW (RecUnfoldC `lang_power`) 0
. |
. ⊢  if n=₂0→ Δ⁰ else ((L↑n-1)⊗L) fi = if n=₂0→ Δ⁰
. |                                    else ((L'↑n-1)⊗L') fi
. | by Eval [`lang_eq`] 0
```

```
. |
. ⊢  ∀l:Alph*.
. |  if n=₂0→ Δ⁰ else ((L↑n-1)⊗L) fi(l)
. |  ⟺
. |  if n=₂0→ Δ⁰ else ((L'↑n-1)⊗L') fi(l)  by Analyze 0
. |
. 9. l : Alph*
. ⊢  if n=₂0→ Δ⁰ else ((L↑n-1)⊗L) fi(l)
. |  ⟺
. |  if n=₂0→ Δ⁰ else ((L'↑n-1)⊗L') fi(l)  by SplitOnConclITE
. |
. |\
. | .....truecase .....
. | 10. n = 0
. | ⊢  Δ⁰(l) ⟺ Δ⁰(l)  by Auto
. |
. .\
. . .....falsecase .....
. . 10. ¬n = 0
. . ⊢  ((L↑n-1)⊗L)(l) ⟺ ((L'↑n-1)⊗L')(l)  by RWH (HypC 8) 0
. . |
. . |\
. . | ⊢  ((L↑n-1)⊗L) ∈ Alph*→Prop  by Fold ‘languages‘ 0
. . |
. . |\
. . | ⊢  ((L'↑n-1)⊗L) ∈ Alph*→Prop  by Fold ‘languages‘ 0
. . |
. . |\
. . | ⊢  ((L'↑n-1)⊗L') ∈ Alph*→Prop  by Fold ‘languages‘ 0
. . |
. . |\
. . | ⊢  ((L'↑n-1)⊗L') ∈ Alph*→Prop  by Fold ‘languages‘ 0
. . |
. . |\
. . | ⊢  ((L'↑n-1)⊗L') = ((L'↑n-1)⊗L')  by Fold ‘languages‘ 0
. . |
. . .\
. . . ⊢  ((L'↑n-1)⊗L)(l) ⟺ ((L'↑n-1)⊗L')(l)
. . .   by RWH (HypC 5) 0 THEN Fold ‘languages‘ 0
END
```