```
assert_term_eq  THM

⊢  ∀a,b:Term. term_eq(a;b) ⟺ a = b
| by Unfold ‘term‘ 0 THEN
D 0 THENA Auto THEN TreeInd (-1) THEN

|    D 0 THENA Auto THEN TreeInd (-1) THEN

|    RecUnfold ‘term_eq‘ 0 THEN Reduce 0
|
|\
| 1. Tree(ts())
| 2. u : Tree(ts())→Type
| 3. a:{v:Tree(ts())| u(v) }→∀b:Tree(ts()). term_eq(a;b) ⟺ a = b
| 4. x : ts()
| 5. Tree(ts())
| 6. u1 : Tree(ts())→Type
| 7. b:{v:Tree(ts())| u1(v) }→term_eq(tree_leaf(x);b)
| |                           ⟺
| |                           tree_leaf(x) = b
| 8. x1 : ts()
| ⊢  (x=x1) ⟺ tree_leaf(x) = tree_leaf(x1)
| | by RWW "assert_ts_eq" 0 THEN Auto
| |
| |\
| | 1. Tree(ts())
| | 2. u : Tree(ts())→Type
| | 3. a:{v:Tree(ts())| u(v) }→∀b:Tree(ts()). term_eq(a;b) ⟺ a = b
| | 4. x : ts()
| | 5. Tree(ts())
| | 6. u1 : Tree(ts())→Type
| | 7. b:{v:Tree(ts())| u1(v) }→term_eq(tree_leaf(x);b)
| |                             ⟺
| |                             tree_leaf(x) = b
| | 8. x1 : ts()
| | 9. x = x1
| | ⊢  tree_leaf(x) = tree_leaf(x1)  by EqCD THEN Auto
| |
| .\
| . 1. Tree(ts())
| . 2. u : Tree(ts())→Type
| . 3. a:{v:Tree(ts())| u(v) }→∀b:Tree(ts()). term_eq(a;b) ⟺ a = b
| . 4. x : ts()
| . 5. Tree(ts())
| . 6. u1 : Tree(ts())→Type
| . 7. b:{v:Tree(ts())| u1(v) }→term_eq(tree_leaf(x);b)
| . |                           ⟺
| . |                           tree_leaf(x) = b
| . 8. x1 : ts()
| . 9. tree_leaf(x) = tree_leaf(x1)
| . ⊢  x = x1
| . | by ApFunToHypEquands ‘z’ Case(z)
| . |                         Case tree_leaf(u) =>
| . |                             u
| . |                         Default => x ts() (-1)
```

```
| .
| . |    THENM Reduce (-1)
| . THEN Auto
| . |
| . .....fun wf .....
| . 1. Tree(ts())
| . 2. u : Tree(ts())→Type
| . 3. a:{v:Tree(ts())| u(v) }→∀b:Tree(ts()). term_eq(a;b) ⟺ a = b
| . 4. x : ts()
| . 5. Tree(ts())
| . 6. u1 : Tree(ts())→Type
| . 7. b:{v:Tree(ts())| u1(v) }→term_eq(tree_leaf(x);b)
| .                              ⟺
| .                              tree_leaf(x) = b
| . 8. x1 : ts()
| . 9. tree_leaf(x) = tree_leaf(x1)
| . 10. z : Tree(ts())
| . ⊢  Case(z)
| .    Case tree_leaf(u) =>
| .        u
| .    Default => x
| .    =
| .    Case(z)
| .    Case tree_leaf(u) =>
| .        u
| .    Default => x  by TreeInd (-1) THEN Reduce 0 THEN Auto
|
|\
| 1. Tree(ts())
| 2. u : Tree(ts())→Type
| 3. a:{v:Tree(ts())| u(v) }→∀b:Tree(ts()). term_eq(a;b) ⟺ a = b
| 4. x : ts()
| 5. Tree(ts())
| 6. u1 : Tree(ts())→Type
| 7. b:{v:Tree(ts())| u1(v) }→term_eq(tree_leaf(x);b)
|                              ⟺
|                              tree_leaf(x) = b
| 8. y1 : {v:Tree(ts())| u1(v) }
| 9. y2 : {v:Tree(ts())| u1(v) }
| ⊢  False ⟺ tree_leaf(x) = tree_node(<y1, y2>)
|   by Auto THEN ApFunToHypEquands ‘z’ Case(z)
|                                     Case x;y =>
|                                         0
|                                     Default => 1 ℤ (-1)
|
|      THENM Reduce (-1)
|
|      THEN Auto THEN TreeInd (-1) THEN Reduce 0 THEN Auto
|
|\
| 1. Tree(ts())
| 2. u : Tree(ts())→Type
| 3. a:{v:Tree(ts())| u(v) }→∀b:Tree(ts()). term_eq(a;b) ⟺ a = b
| 4. y1 : {v:Tree(ts())| u(v) }
| 5. y2 : {v:Tree(ts())| u(v) }
```

```
| 6. Tree(ts())
| 7. u1 : Tree(ts())→Type
| 8. b:{v:Tree(ts())| u1(v) }→term_eq(tree_node(<y1, y2>);b)
|                               ⟺
|                               tree_node(<y1, y2>) = b
| 9. x : ts()
| ⊢  False ⟺ tree_node(<y1, y2>) = tree_leaf(x)
|   by Auto THEN ApFunToHypEquands ‘z’ Case(z)
|                                       Case x;y =>
|                                           0
|                                       Default => 1 ℤ (-1)
|
|      THENM Reduce (-1)
|
|      THEN Auto THEN TreeInd (-1) THEN Reduce 0 THEN Auto
|
.\
. 1. Tree(ts())
. 2. u : Tree(ts())→Type
. 3. a:{v:Tree(ts())| u(v) }→∀b:Tree(ts()). term_eq(a;b) ⟺ a = b
. 4. y1 : {v:Tree(ts())| u(v) }
. 5. y2 : {v:Tree(ts())| u(v) }
. 6. Tree(ts())
. 7. u1 : Tree(ts())→Type
. 8. b:{v:Tree(ts())| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. |                             ⟺
. |                             tree_node(<y1, y2>) = b
. 9. y3 : {v:Tree(ts())| u1(v) }
. 10. y4 : {v:Tree(ts())| u1(v) }
. ⊢  (term_eq(y1;y3)∧₂term_eq(y2;y4))
. |  ⟺
. |  tree_node(<y1, y2>) = tree_node(<y3, y4>)
. | by All (Fold ‘term‘) THEN Auto
. |
. |\
. | 1. Term
. | 2. u : Term→Type
. | 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | 4. y1 : {v:Term| u(v) }
. | 5. y2 : {v:Term| u(v) }
. | 6. Term
. | 7. u1 : Term→Type
. | 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | |                         ⟺
. | |                         tree_node(<y1, y2>) = b
. | 9. y3 : {v:Term| u1(v) }
. | 10. y4 : {v:Term| u1(v) }
. | 11. term_eq(y1;y3)∧₂term_eq(y2;y4)
. | ⊢  tree_node(<y1, y2>) = tree_node(<y3, y4>)
. | | by RW assert_pushdownC (-1) THENA Auto THEN D (-1)
. | |
. | 1. Term
. | 2. u : Term→Type
. | 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | 4. y1 : {v:Term| u(v) }
```

```
. | 5. y2 : {v:Term| u(v) }
. | 6. Term
. | 7. u1 : Term→Type
. | 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | |                      ⟺
. | |                      tree_node(<y1, y2>) = b
. | 9. y3 : {v:Term| u1(v) }
. | 10. y4 : {v:Term| u1(v) }
. | 11. term_eq(y1;y3)
. | 12. term_eq(y2;y4)
. | ⊢  tree_node(<y1, y2>) = tree_node(<y3, y4>)
. | | by Subst y1 = y3 0
. | THENM Subst y2 = y4 0
. |
. | |    THENA (Auto THEN Try (Fold `tapp` 0) THEN Auto)
. | |
. | |\
. | | .....equality .....
. | | 1. Term
. | | 2. u : Term→Type
. | | 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | | 4. y1 : {v:Term| u(v) }
. | | 5. y2 : {v:Term| u(v) }
. | | 6. Term
. | | 7. u1 : Term→Type
. | | 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | |                        ⟺
. | |                        tree_node(<y1, y2>) = b
. | | 9. y3 : {v:Term| u1(v) }
. | | 10. y4 : {v:Term| u1(v) }
. | | 11. term_eq(y1;y3)
. | | 12. term_eq(y2;y4)
. | | ⊢  y1 = y3  by BackThruSomeHyp THEN Auto
. | |
. | |\
. | | .....equality .....
. | | 1. Term
. | | 2. u : Term→Type
. | | 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | | 4. y1 : {v:Term| u(v) }
. | | 5. y2 : {v:Term| u(v) }
. | | 6. Term
. | | 7. u1 : Term→Type
. | | 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | |                        ⟺
. | |                        tree_node(<y1, y2>) = b
. | | 9. y3 : {v:Term| u1(v) }
. | | 10. y4 : {v:Term| u1(v) }
. | | 11. term_eq(y1;y3)
. | | 12. term_eq(y2;y4)
. | | ⊢  y2 = y4  by BackThruSomeHyp THEN Auto
. | |
. | .\
. | . 1. Term
. | . 2. u : Term→Type
```

```
. | . 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | . 4. y1 : {v:Term| u(v) }
. | . 5. y2 : {v:Term| u(v) }
. | . 6. Term
. | . 7. u1 : Term→Type
. | . 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | .                         ⟺
. | .                         tree_node(<y1, y2>) = b
. | . 9. y3 : {v:Term| u1(v) }
. | . 10. y4 : {v:Term| u1(v) }
. | . 11. term_eq(y1;y3)
. | . 12. term_eq(y2;y4)
. | . ⊢  tree_node(<y3, y4>) = tree_node(<y3, y4>)
. | .   by Fold `tapp` 0 THEN Auto
. |
. |\
. | 1. Term
. | 2. u : Term→Type
. | 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | 4. y1 : {v:Term| u(v) }
. | 5. y2 : {v:Term| u(v) }
. | 6. Term
. | 7. u1 : Term→Type
. | 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | |                         ⟺
. | |                         tree_node(<y1, y2>) = b
. | 9. y3 : {v:Term| u1(v) }
. | 10. y4 : {v:Term| u1(v) }
. | 11. tree_node(<y1, y2>) = tree_node(<y3, y4>)
. | ⊢  term_eq(y1;y3)∧₂term_eq(y2;y4)
. | | by RW assert_pushdownC 0 THENA Auto THEN D 0
. |
. | |    THEN BackThruSomeHyp
. | |
. | |\
. | | 1. Term
. | | 2. u : Term→Type
. | | 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | | 4. y1 : {v:Term| u(v) }
. | | 5. y2 : {v:Term| u(v) }
. | | 6. Term
. | | 7. u1 : Term→Type
. | | 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | |                         ⟺
. | |                         tree_node(<y1, y2>) = b
. | | 9. y3 : {v:Term| u1(v) }
. | | 10. y4 : {v:Term| u1(v) }
. | | 11. tree_node(<y1, y2>) = tree_node(<y3, y4>)
. | | ⊢  y1 = y3
. | |   by ApFunToHypEquands `z' Case(z)
. | |                             Case u;v =>
. | |                                 u
. | |                             Default => y1 Term (-1)
. | |
. | |      THENM Reduce (-1)
```

```
. | | THEN Auto
. | |
. | |      THEN TermInd (-1) THEN Reduce 0 THEN Auto
. | |
. | .\
. | . 1. Term
. | . 2. u : Term→Type
. | . 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | . 4. y1 : {v:Term| u(v) }
. | . 5. y2 : {v:Term| u(v) }
. | . 6. Term
. | . 7. u1 : Term→Type
. | . 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. | .                       ⟺
. | .                       tree_node(<y1, y2>) = b
. | . 9. y3 : {v:Term| u1(v) }
. | . 10. y4 : {v:Term| u1(v) }
. | . 11. tree_node(<y1, y2>) = tree_node(<y3, y4>)
. | . ⊢  y2 = y4
. | .   by ApFunToHypEquands ‘z’ Case(z)
. | .                           Case u;v =>
. | .                               v
. | .                           Default => y1 Term (-1)
. | .
. | .      THENM Reduce (-1)
. | . THEN Auto
. | .
. | .      THEN TermInd (-1) THEN Reduce 0 THEN Auto
. |
. |\
. | .....subterm .....
. | 1. Term
. | 2. u : Term→Type
. | 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. | 4. y1 : {v:Term| u(v) }
. | 5. y2 : {v:Term| u(v) }
. | 6. Term
. | 7. u1 : Term→Type
. | 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
. |                       ⟺
. |                       tree_node(<y1, y2>) = b
. | 9. {v:Term| u1(v) }
. | 10. {v:Term| u1(v) }
. | ⊢  tree_node(<y1, y2>) ∈ Term  by Fold ‘tapp‘ 0 THEN Auto
. |
. .\
. . .....subterm .....
. . 1. Term
. . 2. u : Term→Type
. . 3. a:{v:Term| u(v) }→∀b:Term. term_eq(a;b) ⟺ a = b
. . 4. y1 : {v:Term| u(v) }
. . 5. y2 : {v:Term| u(v) }
. . 6. Term
. . 7. u1 : Term→Type
. . 8. b:{v:Term| u1(v) }→term_eq(tree_node(<y1, y2>);b)
```

```
. .                          ⟺
. .                          tree_node(<y1, y2>) = b
. . 9. y3 : {v:Term| u1(v) }
. . 10. y4 : {v:Term| u1(v) }
. . ⊢  tree_node(<y3, y4>) ∈ Term  by Fold ‘tapp‘ 0 THEN Auto
END
```