

课后阅读：用递归函数定义数学性质

我们同样可以利用递归函数定义二叉树的一些性质。

```
Fixpoint tree_le (ub: Z) (t: tree): Prop :=
  match t with
  | Leaf => True
  | Node l k r => tree_le ub l /\ k <= ub /\ tree_le ub r
  end.
```

```
Fixpoint tree_ge (lb: Z) (t: tree): Prop :=
  match t with
  | Leaf => True
  | Node l k r => tree_ge lb l /\ k >= lb /\ tree_ge lb r
  end.
```

这里，`tree_le n t` 表示树中每个节点标号的值都小于等于 `n`，类似的，`tree_ge n t` 表示树中每个节点标号的值都大于等于 `n`。之后我们会用

$$t \subseteq [n, +\infty] \text{ 与 } t \subseteq [-\infty, n]$$

表示 `tree_le n t` 与 `tree_ge n t`。

进一步，我们还可以定义，树中元素根据中序遍历是从小到大排列的 `low2high`，或从大到小排列的 `high2low` 这两条性质。

```
Fixpoint low2high (t: tree): Prop :=
  match t with
  | Leaf => True
  | Node l k r => low2high l /\ l \subseteq [-\infty, k] /\ r \subseteq [k, +\infty] /\ low2high r
  end.
```

```
Fixpoint high2low (t: tree): Prop :=
  match t with
  | Leaf => True
  | Node l k r => high2low l /\ l \subseteq [k, +\infty] /\ r \subseteq [-\infty, k] /\ high2low r
  end.
```

下面证明一些关于它们的有趣性质。我们先试着证明：如果 `t` 中元素中序遍历是从小到大的，那么将其左右翻转后，其中元素的中序遍历是从大到小的。

```

Lemma reverse_low2high: forall t,
  low2high t ->
  high2low (tree_reverse t).
Proof.
  intros.
  induction t.
+ (** 奠基步骤是显然成立的。*)
  simpl. tauto.
+ simpl in H.
  simpl.
  (** 归纳步骤的证明目标是：
    - H: low2high t1 /\ t1 ⊆ [ - ∞ , v ] /\
        t2 ⊆ [ v , + ∞ ] /\ low2high t2
    - IHt1: low2high t1 ->
        high2low (tree_reverse t1)
    - IHt2: low2high t2 ->
        high2low (tree_reverse t2)
    - 结论: high2low (tree_reverse t2) /\
        tree_reverse t2 ⊆ [ v , + ∞ ] /\
        tree_reverse t1 ⊆ [ - ∞ , v ] /\
        high2low (tree_reverse t1)
    这样看来，我们需要一些关于 tree_le 与 tree_ge 的辅助引理。*)
Abort.

```

下面首先证明，如果一棵树中的元素都小于等于 `n`，那么它左右取反后，树中的元素依然都小于等于 `n`。

```

Lemma reverse_le:
  forall n t,
    t ⊆ [ - ∞ , n ] ->
    tree_reverse t ⊆ [ - ∞ , n ].
Proof.
  intros.
  induction t; simpl.
+ tauto.
+ simpl in H.
  tauto.
Qed.

```

其次证明相对称的关于 `tree_ge` 的引理。

```

Lemma reverse_ge:
  forall n t,
    t ⊆ [ n , + ∞ ] ->
    tree_reverse t ⊆ [ n , + ∞ ].
Proof.
  intros.
  induction t; simpl.
+ tauto.
+ simpl in H.
  tauto.
Qed.

```

现在，准备工作已经就绪，可以开始证明 `reverse_low2high` 了。

```

Lemma reverse_low2high: forall t,
  low2high t ->
  high2low (tree_reverse t).
Proof.
  intros.
  induction t; simpl.
+ tauto.
+ simpl in H.
  pose proof reverse_le v t1.
  pose proof reverse_ge v t2.
  tauto.
Qed.

```

最后，我们再定义一个关于两棵树的性质，并证明几个基本结论。

```

Fixpoint same_structure (t1 t2: tree): Prop :=
  match t1, t2 with
  | Leaf, Leaf =>
    True
  | Leaf, Node _ _ _ =>
    False
  | Node _ _ _, Leaf =>
    False
  | Node l1 _ r1, Node l2 _ r2 =>
    same_structure l1 l2 /\ same_structure r1 r2
  end.

```

这个定义说的是：两棵二叉树结构相同，但是每个节点上标号的值未必相同。从这一定义的语法也不难看出，Coq 中允许同时对多个对象使用 `match` 并且可以用下划线省去用不到的 `match` 信息。

下面证明，如果两棵树结构相同，那么它们的高度也相同。

```

Lemma same_structure_same_height: forall t1 t2,
  same_structure t1 t2 ->
  tree_height t1 = tree_height t2.
Proof.
  intros.
  (** 要证明这一结论，很自然的思路是要对 _[t1]_ 做结构归纳证明。这样一来，当 _[t1]_
   为非空树时，归纳假设大致是：_[t1]_ 的左右子树分别与 _[t2]_ 的左右子树结构相
   同，显然，这样的归纳假设可以理解推出最终的结论。*)
  induction t1.
  (** 下面先进行奠基步骤的证明。*)
  + destruct t2.
    - reflexivity.
    - simpl in H.
      tauto.
  + (** 下面进入归纳步骤。然而，通过观察此时的证明目标，我们会发现，当前证明目标与
   我们先前的设想并不一致！我们设想的证明步骤中，归纳假设应当是归于 _[t2]_ 的子
   树的，而非归于 _[t2]_ 本身的。这里的问题在于，当我们执行 _[induction t1]_ 证明
   指令时，_[t2]_ 已经在证明目标的前提中了，这意味着，我们告诉 Coq，要对某个特
   定的 _[t2]_ 完成后续证明。这并不是我们先前拟定的证明思路。*)
Abort.

```

解决这一问题的办法是像我们先前介绍的那样采用加强归纳法。

```

Lemma same_structure_same_height: forall t1 t2,
  same_structure t1 t2 ->
  tree_height t1 = tree_height t2.
Proof.
  intros t1.
  induction t1 as [| l1 IHl v1 r1 IHr]; intros.
+ (** 奠基步骤与原先类似。 *)
  destruct t2.
  - reflexivity.
  - simpl in H.
    tauto.
+ (** 归纳步骤中，归纳假设现在不同了 *)
  destruct t2 as [| l2 v2 r2]; simpl in H.
  - tauto.
  - destruct H as [H1 Hr].
    (** 现在我们可以将归纳假设作用在_[t2]_的子树上了。 *)
    pose proof IHl l2 H1.
    pose proof IHr r2 Hr.
    simpl.
    lia.
Qed.

```

习题 1. 下面的证明留作习题。

```

Theorem same_structure_trans: forall t1 t2 t3,
  same_structure t1 t2 ->
  same_structure t2 t3 ->
  same_structure t1 t3.
(* 请在此处填入你的证明，以_[Qed]_结束。 *)

```