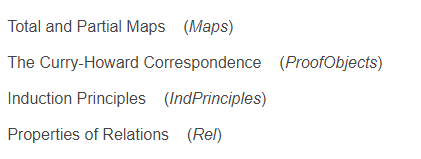
考试范围：

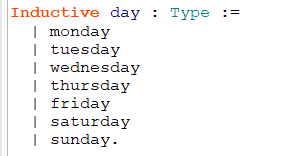
\*不考\*



一直到Imp.v

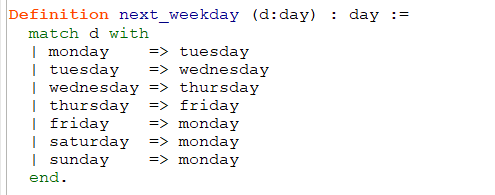
1. Basics.v

类型定义：



**用Definition定义函数**

定义一个参数类型为day，返回类型为day的函数：

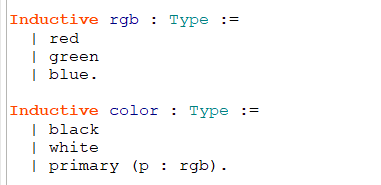




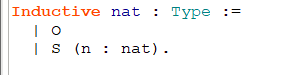
布尔函数：negb, andb, orb, xor(异或)

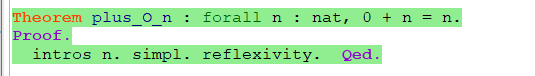
Check: 查看表达式类型

复合类型的定义：

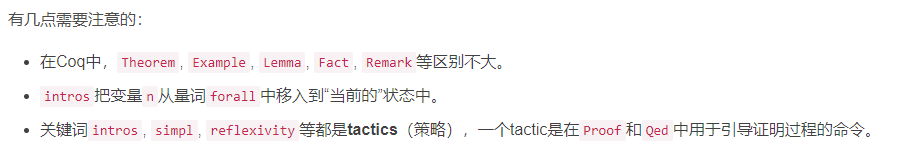


归纳定义：允许类型的构造器接收与它同类型的参数，比如自然数(natural number, nat)的定义：

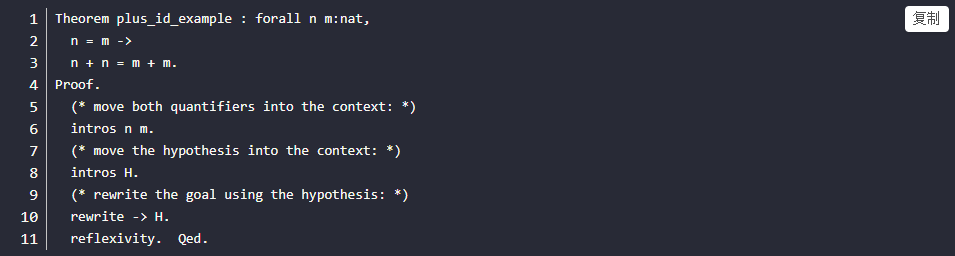




上面的例子中 ，simpl不是必需的，它只是为了方便查看简化的中间过程；reflexivity在检查等号两边是否相等前可以自动完成一些简化工作。reflexivity可以打开(unfolding)一些项，把这些项替换成定义时写在右边的东西。



Rewrite：

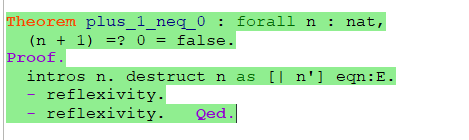


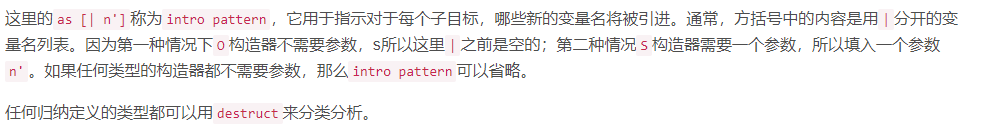
上面的->读作“隐含/暗示”，n = m为假设。因为n和m是任意数字，所以不能简单地用简化来证明该定理。在n = m的假设下，我们把目标语句中的n替换成m即可完成证明。这种tactic叫做**rewrite**。

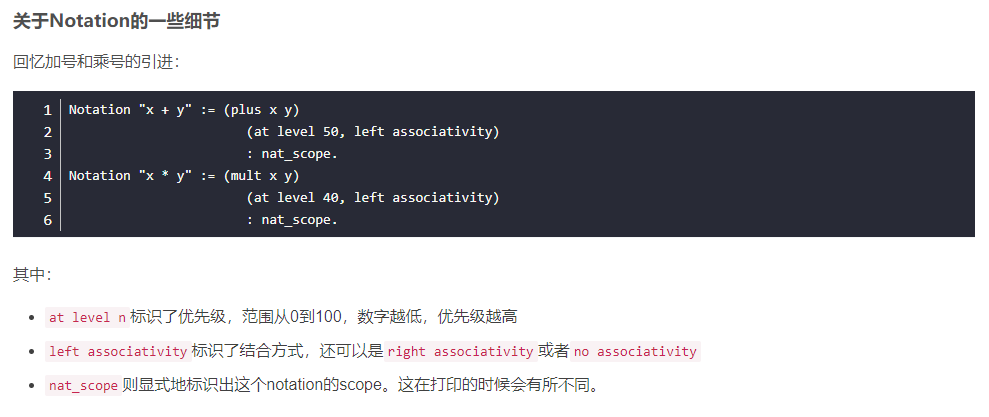
这里rewrite -> H.表示通过把假设等式左边n替换成右边m来完成rewrite。如果要把等式右边的m换成左边的n，可以用<-。

另外，rewrite的对象除了是假设，也可以说之前证明过的定理。如果引用的定理中含有量词，Coq就会尝试匹配当前目标来将其实例化

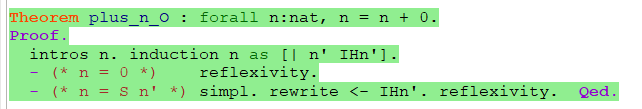
Destruct： 根据某个变量的构造器进行分类







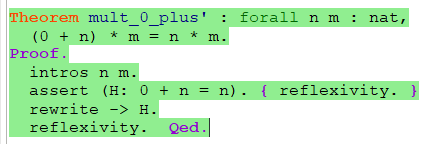
1. Induction.v



Coq中提供了一个叫induction的tactic，它把上述定理的证明分成两个子目标：

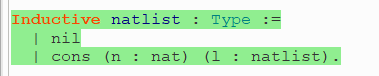
1. 证明P(O)成立
2. 证明P(n')成立可以推导出P(S n')

用assert证明局部定理：



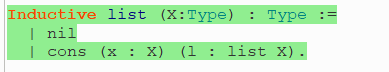
1. Lists.v

Natlist 定义：

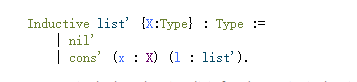


1. Poly.v

归纳定义list（type->type）



隐式参数：

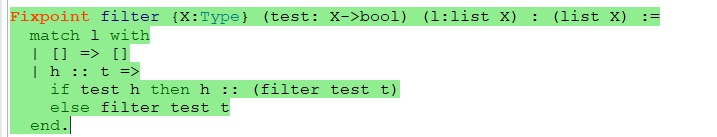


高阶函数：



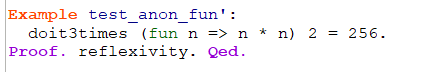
参数f本身为一个函数，这里将f执行3次

过滤器：

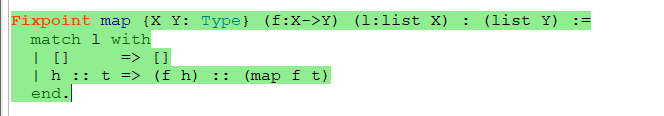


保留列表list中满足test的元素

使用fun 构造函数，不需给函数命名



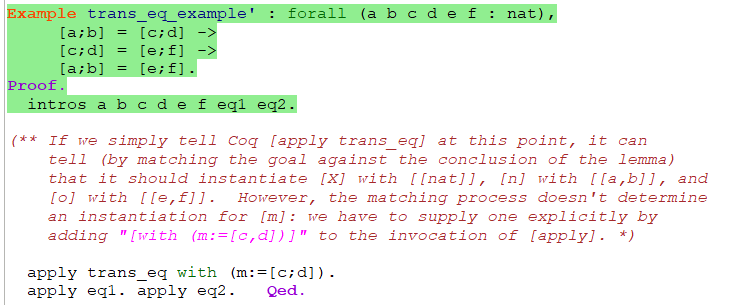
map：对列表中所有元素执行同一操作



1. Tactics.v

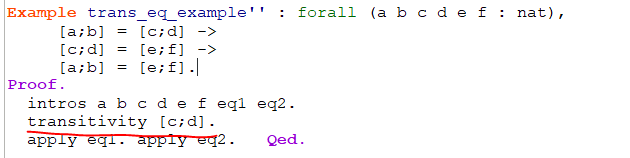
apply: rewrite 和 reflexivity.

apply with:

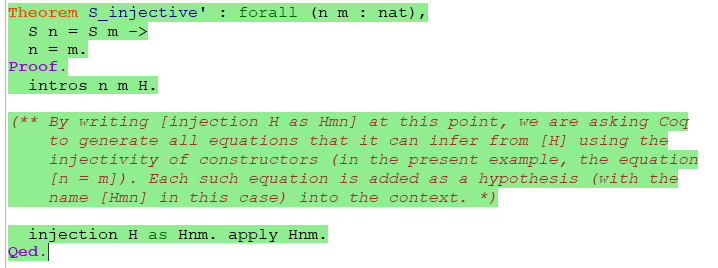


括号中m可省略，括号中内容告诉coq如何进行匹配。

也可以用transitivity



injection:



H: S m = S n

Hmn: m = n

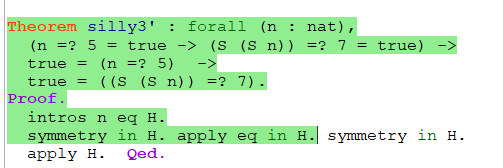
通过[injection H]语句，Coq会生成它可以依据构造函数的单射性推导出的所有方程。每一个这样的方程都是作为目标的前提加入的

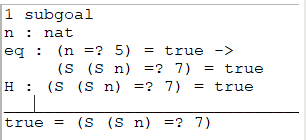
discriminate:

如果证明过程中某个荒谬的假设成立，那么推出的任何结论也是成立的

apply L in H:

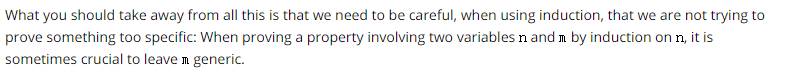
换句话说，在H中应用L给我们一种“前向推理”的形式：从X→Y和一个与X匹配的假设，它产生一个与Y匹配的假设。相反，应用L是“向后推理”：它说如果 我们知道X→Y，我们试图证明Y，证明X就足够了。

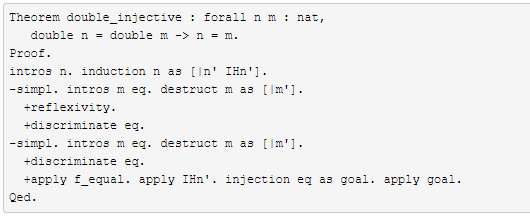




易变的归纳性假设：

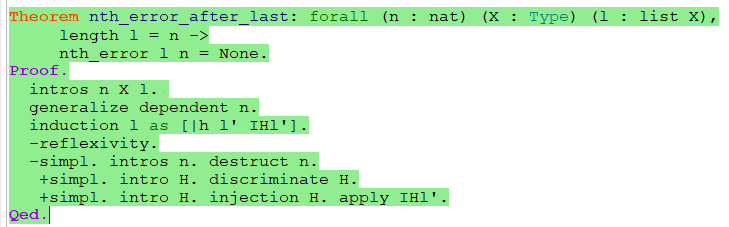
有两个变量n m时， 先只intros n，induction n，然后在两种情况下分别intros m， destruct m





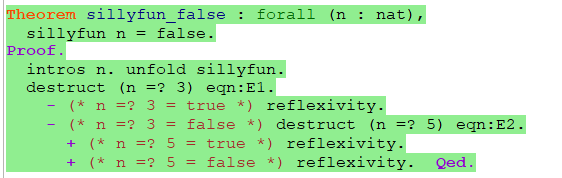
generalize dependent n.

将n返回到goal中（相当于撤回intros n操作）

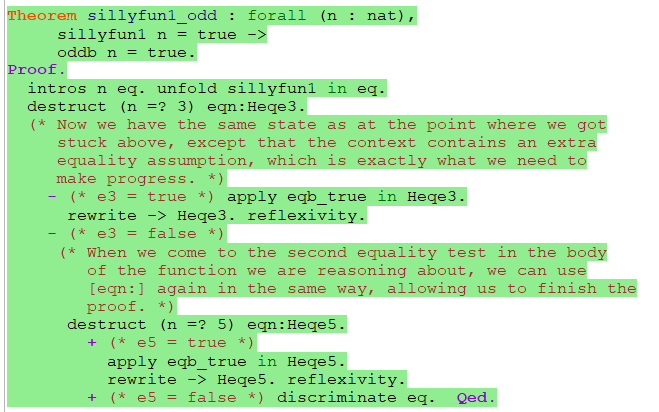


unfold： 展开某个函数或定理

在复合表达式中使用destruct：



[destruct]策略中的"eqn:"部分是可选的，但大多数时候都会选择加上这个部分仅仅是为文档化，但是大多数的Coq证明都会忽略这个部分。  
此外，需要注意的是，当对复合表达式使用[destruct]策略时，由eqn部分记录的信息有时会是决定性的：如果忽略这部分信息，那么[destruct]策略可能会删除掉在完整证明的过程中所需要的信息。  
比如：



总结：

