课后阅读：布尔类型表达式的指称语义

1 布尔表达式的语义

对于任意布尔表达式 e，我们规定它的语义 $[\![e]\!]$ 是一个程序状态到真值的函数，表示表达式 e 在各个程序状态上的求值结果。

- $[\![\text{TRUE}]\!](s) = \mathbf{T}$
- $[\![\text{FALSE}]\!](s) = \mathbf{F}$
- $[\![e_1 < e_2]\!](s)$ 为真当且仅当 $[\![e_1]\!](s) < [\![e_2]\!](s)$
- $[\![e_1 \&\& e_2]\!](s) = [\![e_1]\!](s)$ and $[\![e_2]\!](s)$
- $[\![!e_1]\!](s) = \text{not } [\![e_1]\!](s)$

在 Coq 中可以如下定义：

```
Definition true_sem: state -> bool :=
  fun s => true.
```

```
Definition false_sem: state -> bool :=
  fun s => false.
```

```
Definition lt_sem (D1 D2: state -> Z):
  state -> bool :=
  fun s =>
    if Z_lt_dec (D1 s) (D2 s)
    then true
    else false.
```

```
Definition and_sem (D1 D2: state -> bool):
  state -> bool :=
  fun s => andb (D1 s) (D2 s).
```

```
Definition not_sem (D: state -> bool):
  state -> bool :=
  fun s => negb (D s).
```

```
Fixpoint eval_expr_bool (e: expr_bool): state -> bool :=
  match e with
  | ETrue =>
      true_sem
  | EFalse =>
      false_sem
  | ELt e1 e2 =>
      lt_sem (eval_expr_int e1) (eval_expr_int e2)
  | EAnd e1 e2 =>
      and_sem (eval_expr_bool e1) (eval_expr_bool e2)
  | ENot e1 =>
      not_sem (eval_expr_bool e1)
  end.
```

与整数类型表达式的行为等价定义一样，我们也可以用函数相等定义布尔表达式行为等价。

```
Definition bequiv (e1 e2: expr_bool): Prop :=
  (⟦ e1 ⟧ == ⟦ e2 ⟧)%func.
```

下面先证明三个语义算子 `lt_sem` 、`and_sem` 与 `not_sem` 能保持函数相等，再利用函数相等的性质证明布尔表达式行为等价的性质。

```
#[export] Instance lt_sem_congr:
  Proper (func_equiv _ _ ==>
          func_equiv _ _ ==>
          func_equiv _ _) lt_sem.
```

```
#[export] Instance and_sem_congr:
  Proper (func_equiv _ _ ==>
          func_equiv _ _ ==>
          func_equiv _ _) and_sem.
```

```
#[export] Instance not_sem_congr:
  Proper (func_equiv _ _ ==> func_equiv _ _) not_sem.
```

```
#[export] Instance bequiv_equiv: Equivalence bequiv.
```

```
#[export] Instance ELt_congr:
  Proper (iequiv ==> iequiv ==> bequiv) ELt.
```

```
#[export] Instance EAnd_congr:
  Proper (bequiv ==> bequiv ==> bequiv) EAnd.
```

```
#[export] Instance ENot_congr:
  Proper (bequiv ==> bequiv) ENot.
```