



软件科学基础

# STLCPROP: Properties of STLC TYPECHECKING: A Typechecker for STLC

熊英飞  
北京大学



# Progress

**Theorem progress** : forall t T,  
empty |-- t \in T ->  
value t \ / exists t', t --> t'.

- 证明概要：在 |-- t \in T 上做归纳
  - 不可能是 T\_Var
  - T\_True, T\_False 和 T\_Abs 的时候 t 都是 value
  - T\_App 的时候，t 为 t1 t2，根据归纳假设
    - t1 或者 t2 能往下约简，则整体可以往下约简
    - t1 和 t2 都是 value，因为 t1 是函数类型，则 t1 必然是 lambda 抽象，所以根据 ST\_AppAbs 可以往下约简
  - T\_If 的时候，t 为 if t1 then t2 else t3，根据归纳假设
    - 如果 t1 是 value，则 t1 为 true 或者 false，整体可以约简
    - 如果 t1 可以往下约简，整体可以往下约简



# Preservation

**Theorem** `preservation` : forall t t' T,  
 empty |-- t \in T ->  
 t --> t' ->  
 empty |-- t' \in T.

- 因为需要对application进行分析，即需要保证替换不影响类型，先证明两个引理。



# 弱化引理

```
Lemma weakening_empty : forall Gamma t T,  
  empty |-- t \in T ->  
  Gamma |-- t \in T.
```

- 证明思路：在推导关系上做归纳，将归纳假设应用到目标上



# 替换引理

Lemma substitution\_preserves\_typing : forall

$$\begin{aligned} & \Gamma \ x \ U \ t \ v \ T, \\ & \ x \ \vdash \ U \ ; \ \Gamma \ \vdash \ t \ \text{in} \ T \ \rightarrow \\ & \ \text{empty} \ \vdash \ v \ \text{in} \ U \ \rightarrow \\ & \ \Gamma \ \vdash \ [x:=v]t \ \text{in} \ T. \end{aligned}$$

## • 证明概要：在t上做归纳

- 如果t是变量且为x，则U=T，用归纳假设和弱化引理可以证明
- 如果t是变量且不为x，则替换不改变任何内容
- 如果t是 $\lambda y:S, t_0$ ，则 $T=S \rightarrow T_0$ 且 $y \vdash S; x \vdash U; \Gamma \vdash t_0 \text{ in } T_0$ 。同时有归纳假设 $\forall \Gamma'$ ， $x \vdash U; \Gamma' \vdash t_0 \text{ in } T_0 \rightarrow \Gamma' \vdash [x:=v]t_0 \text{ in } T_0$ 。
  - 如果 $x=y$ ，则替换之后还是t，根据T-Abs我们需要证明 $y \vdash S; \Gamma \vdash t_0 \text{ in } T_0$ 。上述类型假设变为 $y \vdash S; y \vdash U; \Gamma \vdash t_0 \text{ in } T_0$ 。二者等价
  - 如果 $x \neq y$ ，则需要证明 $y \vdash S; \Gamma \vdash [x:=v]t_0 \text{ in } T_0$ 。令归纳假设中 $\Gamma' = y \vdash S; \Gamma$ 可得
- 其他情况应用归纳假设可得。



# 证明Preservation

**Theorem** preservation : forall t t' T,  
empty |-- t \in T ->  
t --> t' ->  
empty |-- t' \in T.

- 在 |--t \in T上做归纳
  - T\_Var, T\_Abs, T\_True, T\_False的情况都不会往下计算
  - T\_App的情况，则t=t1 t2
    - 如果t1或t2可以往下约简，则应用归纳假设可得
    - 如果t1和t2都是value，则应用替换引理可得
  - T\_If的情况，则t=if t1 then t2 else t3
    - 如果t1可以往下约简，应用归纳假设可得
    - 如果t1不能往下约简，则整体约简为t2或者t3，类型保持



# Preservation的逆是否成立

```
forall t t' T,  
  empty |-- t' \in T ->  
  t --> t' ->  
  empty |-- t \in T.
```

- 不成立，因为类型有错的项可能规约成类型正确的项，如 $(\lambda x:\text{Bool}, x) (\lambda x:\text{Bool} x)$



# 类型系统正确性

**Definition** `stuck` ( $t:tm$ ) : `Prop` :=  
(`normal_form step`)  $t$  /\  $\sim$  `value`  $t$ .

**Corollary** `soundness` : `forall`  $t$   $t'$   $T$ ,  
`empty` |--  $t$  \in  $T$  ->  
 $t$  -->\*  $t'$  ->  
 $\sim$ (`stuck`  $t'$ ).





# 类型唯一性

```
Theorem unique_types : forall Gamma e T T',  
  Gamma |-- e \in T ->  
  Gamma |-- e \in T' ->  
  T = T'.
```

证明留作作业



# 类型检查

- 类型唯一性说明has\_type关系是一个函数
- 能否在Coq写出该函数，实现自动检查类型正确性？



# 辅助函数：判断类型等价

```
Fixpoint eqb_ty (T1 T2:ty) : bool :=
  match T1,T2 with
  | <{ Bool }> , <{ Bool }> =>
    true
  | <{ T11->T12 }>, <{ T21->T22 }> =>
    andb (eqb_ty T11 T21) (eqb_ty T12 T22)
  | _,_ =>
    false
  end.
```



# 类型检查函数

```
Fixpoint type_check (Gamma : context) (t : tm) : option ty :=
  match t with
  | tm_var x =>
    Gamma x
  | <{\x:T2, t1}> =>
    match type_check (x |-> T2 ; Gamma) t1 with
    | Some T1 => Some <{T2->T1}>
    | _ => None
    end
  | <{t1 t2}> =>
    match type_check Gamma t1, type_check Gamma t2 with
    | Some <{T11->T12}>, Some T2 =>
      if eqb_ty T11 T2 then Some T12 else None
    | _,_ => None
    end
  end
```



# 类型检查函数

```
| <{true}> =>  
  Some <{Bool}>  
| <{false}> =>  
  Some <{Bool}>  
| <{if guard then t else f}> =>  
  match type_check Gamma guard with  
  | Some <{Bool}> =>  
    match type_check Gamma t, type_check Gamma f with  
    | Some T1, Some T2 =>  
      if eqb_ty T1 T2 then Some T1 else None  
    | _, _ => None  
    end  
  | _ => None  
  end  
end.
```



# 类型检查函数的性质

```
Theorem type_checking_sound : forall Gamma t T,  
  type_check Gamma t = Some T -> has_type Gamma t T.
```

```
Theorem type_checking_complete : forall Gamma t T,  
  has_type Gamma t T -> type_check Gamma t = Some T.
```

证明思路：在 $t$ 或者 $has\_type$ 上做归纳，对应调用另外一边函数或者 $constructor$ 即可



# 练习

- 如果给STLC添加了如下操作语义规则和类型推导规则，progress和preservation是否还成立？

$$\frac{}{t \rightarrow \text{zap}} \quad (\text{ST\_Zap})$$

$$\frac{}{\text{Gamma} \vdash \text{zap} \in \mathbb{T}} \quad (\text{T\_Zap})$$



# 作业

- 完成STLCPROP的progress\_from\_term\_ind和unique\_types
  - 请使用最新英文版教材