软件理论基础与实践

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
Gamma x U t v T,
  x |-> U ; Gamma |- t \in T ->
  empty |- v \in U   ->
  Gamma |- [x:=v]t \in T.
```

- 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\y:S, t0，则T=S->T1且y|->S; x|->U; Gamma |- t0 in T1。同时有归纳假设∀Gamma’, x|->U; Gamma’ |- t0 in T0 → Gamma’ |- [x:=v]t0 in T0.
    - 如果x=y，则替换之后还是t，根据T-Abs我们需要证明y|->S; Gamma |- t0 in T1。上述类型假设变为y|->S; y|->U; Gamma |- t0 in T1。二者等价
    - 如果x≠y，则我们需要证明y|->S; Gamma |- [x:=v]t0 in T1。令归纳假设中Gamma’=y|-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(\x:Bool, x) (\x:Bool x)

类型系统正确性

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

Corollary soundness : forall t t' T,
  empty |- t \in T ->
  t -->* t' ->
  ~(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
```

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?

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

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

作业

- 完成STLCPROP的progress_from_term_ind和unique_types
  - 请使用最新英文版教材