



软件科学基础

Types: Type Systems

熊英飞
北京大学



课程进展

- Coq ✓
- 数理逻辑基础 ✓
- 形式语义 ✓
- 类型系统



什么是类型系统?

- 类型系统是一种用来自动和高效证明程序没有某类错误的语法手段
 - 没有某类错误
 - C的类型系统: 保证变量之间没有跨类型的错误赋值
 - Java的异常处理机制: 保证程序执行中没有不被处理的异常 (runtimeException除外)
 - 不能证明有某类错误
 - 通不过类型检查不代表运行时有可能出现错误
 - 语法手段
 - 通常需要程序员在程序中加上额外标记
 - 自动
 - 加上标记之后证明是自动的
 - 高效
 - 证明过程不能对编译速度有太大影响



类型系统的重要性

- 现代程序设计语言都有较为复杂的类型系统
- 类型系统设计和语言其他部分设计通常同时进行
- 类型理论是软件理论的核心组成部分
- 类型系统有很多作用
 - 编译时错误检测
 - 抽象：类型系统是对程序行为的抽象
 - 文档：类型系统帮助更好的撰写程序
 - 效率：加上类型系统的程序可能会更高效



类型系统的实现思想

- 对程序状态进行分类
- 程序分析角度——类型定义了系统状态的抽象域和抽象域上要检查的断言



带类型的算术表达式：语法

- 之前我们在语法上严格区分了aexp和bexp
 - 不可能写出if(a+1) then 1 else 2
- 为了体现类型的作用，我们在语法上混用自然数和布尔值

- ```
t ::= true
 | false
 | if t then t else t
 | 0
 | succ t
 | pred t
 | iszero t
```

- 但在更复杂的情况中（如有变量），类型系统的作用很难被语法取代



# 语法元素的Coq定义

```
Inductive tm : Type :=
 | tru : tm
 | fls : tm
 | ite : tm -> tm -> tm -> tm
 | zro : tm
 | scc : tm -> tm
 | prd : tm -> tm
 | iszro : tm -> tm.
```



# 定义值

```
Inductive bvalue : tm -> Prop :=
 | bv_true : bvalue <{ true }>
 | bv_false : bvalue <{ false }>.
```

```
Inductive nvalue : tm -> Prop :=
 | nv_0 : nvalue <{ 0 }>
 | nv_succ : forall t, nvalue t -> nvalue <{ succ t }>.
```

```
Definition value (t : tm) := bvalue t \/ nvalue t.
```





# 小步法语义

$$\frac{}{\text{if true then } t_1 \text{ else } t_2 \rightarrow t_1} \quad (\text{ST\_IfTrue})$$
$$\frac{}{\text{if false then } t_1 \text{ else } t_2 \rightarrow t_2} \quad (\text{ST\_IfFalse})$$
$$\frac{t_1 \rightarrow t_1'}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightarrow \text{if } t_1' \text{ then } t_2 \text{ else } t_3} \quad (\text{ST\_If})$$



# 小步法语义

$$\frac{t_1 \rightarrow t_1'}{\text{succ } t_1 \rightarrow \text{succ } t_1'} \quad (\text{ST\_Succ})$$

$$\frac{}{\text{pred } 0 \rightarrow 0} \quad (\text{ST\_Pred0})$$

$$\frac{\text{numeric value } v}{\text{pred } (\text{succ } v) \rightarrow v} \quad (\text{ST\_PredSucc})$$

$$\frac{t_1 \rightarrow t_1'}{\text{pred } t_1 \rightarrow \text{pred } t_1'} \quad (\text{ST\_Pred})$$



# 小步法语义

$$\frac{}{\text{iszero } 0 \rightarrow \text{true}} \quad (\text{ST\_IsZero0})$$

$$\frac{\text{numeric value } v}{\text{iszero } (\text{succ } v) \rightarrow \text{false}} \quad (\text{ST\_IszeroSucc})$$

$$\frac{t_1 \rightarrow t_1'}{\text{iszero } t_1 \rightarrow \text{iszero } t_1'} \quad (\text{ST\_Iszero})$$



# 小步法语义的Coq定义

Reserved Notation "t '-->' t'" (at level 40).

```
Inductive step : tm -> tm -> Prop :=
| ST_IfTrue : forall t1 t2,
 <{ if true then t1 else t2 }> --> t1
| ST_IfFalse : forall t1 t2,
 <{ if false then t1 else t2 }> --> t2
| ST_If : forall c c' t2 t3,
 c --> c' ->
 <{ if c then t2 else t3 }> --> <{ if c' then t2 else t3 }>
| ST_Succ : forall t1 t1',
 t1 --> t1' ->
 <{ succ t1 }> --> <{ succ t1' }>
```



# 小步法语义的Coq定义

```
| ST_Pred0 :
 <{ pred 0 }> --> <{ 0 }>
| ST_PredSucc : forall v,
 nvalue v ->
 <{ pred (succ v) }> --> v
| ST_Pred : forall t1 t1',
 t1 --> t1' ->
 <{ pred t1 }> --> <{ pred t1' }>
| ST_Iszero0 :
 <{ iszero 0 }> --> <{ true }>
| ST_IszeroSucc : forall v,
 nvalue v ->
 <{ iszero (succ v) }> --> <{ false }>
| ST_Iszero : forall t1 t1',
 t1 --> t1' ->
 <{ iszero t1 }> --> <{ iszero t1' }>
```

where "t '-->' t'" := (step t t').



# 标准型

- Strong Progress性质要求标准型一定是值
- 但现在的标准型却不一定是值，如
  - `scc tru`
- 称上面的项是卡住的项

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

```
Lemma some_term_is_stuck :
 exists t, stuck t.
```



# 类型

- 类型帮助我们避免卡住的项

```
Inductive ty : Type :=
 | Bool : ty
 | Nat : ty.
```



# 类型关系

- 将表达式关联到计算结果的类型上

$$\frac{}{\vdash \text{true} \in \text{Bool}} \quad (\text{T\_True})$$

$$\frac{}{\vdash \text{false} \in \text{Bool}} \quad (\text{T\_False})$$

$$\frac{\vdash t_1 \in \text{Bool} \quad \vdash t_2 \in T \quad \vdash t_3 \in T}{\vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \in T} \quad (\text{T\_If})$$

$$\frac{}{\vdash 0 \in \text{Nat}} \quad (\text{T\_0})$$

$$\frac{\vdash t_1 \in \text{Nat}}{\vdash \text{pred } t_1 \in \text{Nat}} \quad (\text{T\_Pred})$$

$$\frac{\vdash t_1 \in \text{Nat}}{\vdash \text{succ } t_1 \in \text{Nat}} \quad (\text{T\_Succ})$$

$$\frac{\vdash t_1 \in \text{Nat}}{\vdash \text{iszero } t_1 \in \text{Bool}} \quad (\text{T\_Iszero})$$





# 类型关系的Coq定义

Reserved Notation "'|--' t '\in' T" (at level 40).

```
Inductive has_type : tm -> ty -> Prop :=
| T_True :
 |-- <{ true }> \in Bool
| T_False :
 |-- <{ false }> \in Bool
| T_If : forall t1 t2 t3 T,
 |-- t1 \in Bool ->
 |-- t2 \in T ->
 |-- t3 \in T ->
 |-- <{ if t1 then t2 else t3 }> \in T
| T_0 :
 |-- <{ 0 }> \in Nat
```



# 类型关系的Coq定义

```
| T_Succ : forall t1,
 |-- t1 \in Nat ->
 |-- <{ succ t1 }> \in Nat
| T_Pred : forall t1,
 |-- t1 \in Nat ->
 |-- <{ pred t1 }> \in Nat
| T_Iszero : forall t1,
 |-- t1 \in Nat ->
 |-- <{ iszero t1 }> \in Bool
```

```
where "'|--' t '\in' T" := (has_type t T).
```



# 进展性 & 保持性

- 进展性：类型正确的项如果不是值，就一定能往下约简
- 保持性：项往下约减之后还保持同样的类型
- 两个属性联合作用保证了项不会卡住



# 进展性Progress

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

作为对比，上一章strong progress属性是对所有term都成立

证明思路：根据推导出 $t \in T$ 关系的类型规则做归纳证明。  
证明需要用到如下lemma

```
Lemma bool_canonical : forall t,
 |-- t \in Bool -> value t ->
 bvalue t.
```

```
Lemma nat_canonical : forall t,
 |-- t \in Nat -> value t ->
 nvalue t.
```



# 保持性Preservation

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

证明思路：还是根据推导出 $t \in T$ 关系的类型规则做归纳证明



# 类型正确性

**Definition** `multistep` := (multi step).

**Notation** "`t1 '--->*' t2`" := (multistep t1 t2) (at level 40).

**Corollary** `soundness` : forall t t' T,

|-- t \in T ->

t --->\* t' ->

~(stuck t').

**Proof.**

intros t t' T HT P. induction P; intros [R S].

- apply progress in HT. destruct HT; auto.

- apply IHP.

+ apply preservation with (t := x); auto.

+ unfold stuck. split; auto.

**Qed.**



# 作业

- 完成Types中standard非optional并不属于Additional Exercises的5道习题
  - 请使用最新英文版教材