Agda佐恩引理

HoTT + AC → Zorn

oCaU 2023.07.15

目录

- 为什么?
- 前置知识
  - AC → LEM → PropResizing
  - 序理论概念:偏序,无界,最大元...
- AC → Zorn
  - 链集的链
  - 归纳构造“塔”

为什么？
动机

- 因为没有被以下形式化
  - HoTT / UF
  - Cubical
- 学习HoTT内经典数学概念的正确表述
- 学习如何在 Cubical Agda 中处理 set level 的概念和推理

AC → LEM

Diaconescu 定理

任意集合间的满射都**存在**右逆

```
SurjectionHasRightInverse : (ℓ ℓ' : Level) → Type _
SurjectionHasRightInverse ℓ ℓ' = {A : Type ℓ} {B : Type ℓ'} {f : A → B} → isSet A → isSet B →
  isSurjection f → ∃[ g ∈ (B → A) ] section f g
```

↓

任意命题可判定

```
LEM : (ℓ : Level) → Type (ℓ-suc ℓ)
LEM ℓ = {P : Type ℓ} (Pprop : isProp P) → Dec P
```

AC → LEM

Diaconescu 定理

商掉布尔类型上的等价关系 ~

布尔类型到该商集有满射

于是存在右逆 g

判定 g [true] 是否等于 g [false]

```
_~_ : Rel Bool Bool ℓ
true  ~ true  = Unit*
false ~ false = Unit*
_     ~ _     = P
```

```
isEquivRel~ : isEquivRel _~_
isEquivRel~ = equivRel isRefl~ isSym~ isTrans~

RightInverse : Type ℓ
RightInverse = Σ[ g ∈ (Bool / _~_ → Bool) ] section [_] g

RightInverse→DecP : RightInverse → Dec P
RightInverse→DecP (g , sec) with g [ true ] ≟ g [ false ]
... | yes p = yes $ effective isPropValued~ isEquivRel~ _ _ $
  sym (sec [ true ]) • cong [_] p • sec [ false ]
... | no ¬p = no $ ¬p ∘ cong g ∘ eq/ _ _
```

LEM → PR

排中律蕴含命题宇宙调整

- 任何宇宙的命题都可以可判定
- 那么都可根据判定出来的真值映射到布尔值
- 该映射是同构

```
hPropIsoBool : ∀ ℓ → Iso (hProp ℓ) Bool
hPropIsoBool ℓ = iso to from to∘from from∘to where
  to : (hProp ℓ) → Bool
  to P with em ⦃ P .snd _ _ ⦄
  ... | yes _ = true
  ... | no  _ = false

  from : Bool → (hProp ℓ)
  from true = ⊤
  from false = ⊥

  to∘from : (b : Bool) → to (from b) ≡ b
  to∘from true  with em {ℓ} ⦃ ⊤ .snd _ _ ⦄
  ... | yes _ = refl
  ... | no ¬p = ⊥.rec $ ¬p tt*
  to∘from false with em {ℓ} ⦃ ⊥ .snd _ _ ⦄
  ... | no  _ = refl

  from∘to : (P : hProp ℓ) → from (to P) ≡ P
  from∘to P with em ⦃ P .snd _ _ ⦄
  ... | yes p = hPropExt $ →: (λ _ → p) ←: (λ _ → tt*)
  ... | no ¬p = hPropExt $ →: (λ ()) ←: (λ p → lift $ ¬p p)
```

序理论概念

偏序

偏序

如果 `R` 取值到命题, 并且满足自反, 反对称和传递性, 则称 `R` 是偏序 (partial order).

```
isPo : Type _
isPo = isPropValued R ∧ isRefl R ∧ isAntisym R ∧ isTrans R
```

如果 `R` 是偏序且 `U` 是集合, 则称 `U` 为偏序集.

```
isPoset : Type _
isPoset = isSet U ∧ isPo
```

序理论概念

无界

无界

我们又用中缀符号 `≤` 表示 `R` 关系.

```
private _≤_ = R
```

我们说 `U` 在 `R` 关系下是无界的, 当且仅当从任意 `x : U` 都能得到一个 `y : U` 严格大于 `x`.

```
unbound : Type _
unbound = ∀ x → Σ[ y ∈ U ] x ≤ y ∧ ¬ x ≡ y
```

我们说 `U` 在 `R` 关系下是后继的, 当且仅当它是无界的, 且见证无界的那个 `y` 刚好比 `x` 大, 也就是说它们之间没有其他元素.

```
successive : Type _
successive = ∀ x → Σ[ y ∈ U ] x ≤ y ∧ (¬ x ≡ y) ∧ ∀ z → x ≤ z → z ≤ y → z ≡ x ∨ z ≡ y
```

序理论概念

最大元

最大元

对任意大于等于 `m` 的元素, 如果它其实都等于 `m`, 那么称 `m` 是 `U` 的最大元.

```
maximum : U → Type _
maximum m = ∀ x → m ≤ x → m ≡ x
```

注意无界与存在最大元是不相容的.

幂集

到hProp的函数

```
𝒫 : Type ℓ → (ℓ' : Level) → Type (ℓ-max ℓ (ℓ-suc ℓ'))
𝒫 X ℓ' = X → hProp ℓ'

isSet𝒫 : isSet (𝒫 X ℓ)
isSet𝒫 = isSetΠ λ x → isSetHProp
```

```
-------------------------------
-- Special sets

-- Empty set

∅ : 𝒫 X ℓ-zero
∅ = λ _ → ⊥

∅* : 𝒫 X ℓ
∅* = λ _ → ⊥* , isProp⊥*

-- Universal set

U : 𝒫 X ℓ-zero
U = λ _ → ⊤

U* : 𝒫 X ℓ
U* = λ _ → Unit* , isPropUnit*

-------------------------------
-- Membership

infix 5 _∈_ _∉_ _⊆_

_∈_ : X → 𝒫 X ℓ → Type _
x ∈ A = ⟨ A x ⟩

_∉_ : X → 𝒫 X ℓ → Type _
x ∉ A = ¬ ⟨ A x ⟩
```

序理论概念

链

链

现在, 考虑 `U` 的子集 `A`, 如果其中的任意两个元素都可以比较大小, 我们就说 `A` 是链, 也叫 `U` 的全序子集.

```
isChain : 𝒫 U ℓ → Type _
isChain A = ∀ x y → x ∈ A → y ∈ A → x ≤ y ∨ y ≤ x
```

注意 `∨` 是和类型 `⊎` 的命题截断, 从而保证了"某某是链"是一个命题. 后面要用到这一性质.

```
isPropIsChain : isProp (isChain A)
isPropIsChain = isPropΠ2 λ _ _ → isPropΠ2 λ _ _ → squash₁
```

序理论概念

上界

上界

给定 `A` 和 `ub : U`, 如果 `ub` 比 `A` 的任意元素都要大, 则称 `ub` 是 `A` 的上界. 注意上界不一定在 `A` 中.

```
upperBound : 𝒫 U ℓ → U → Type _
upperBound A ub = ∀ x → x ∈ A → x ≤ ub
```

由以上定义, “所有链都可以找到上界”表述如下.

```
allChainHasUb = ∀ {ℓ} (A : 𝒫 U ℓ) → isChain A → Σ[ ub ∈ U ] upperBound A ub
```

序理论概念

上确界

上确界

给定 `A` 和 `sup : U`, 如果 `sup` 是 `A` 的最小上界, 则称 `sup` 是 `A` 的上确界. 注意上确界也不一定在 `A` 中.

```
supremum : 𝒫 U ℓ → U → Type _
supremum A sup = upperBound A sup ∧ ∀ ub → upperBound A ub → sup ≤ ub
```

由以上定义, “所有链都可以找到上确界”表述如下.

```
allChainHasSup = ∀ {ℓ} (A : 𝒫 U ℓ) → isChain A → Σ[ sup ∈ U ] supremum A sup
```

Σ与∃的区别

Σ　Simga类型　　找到，取到，得到

∃　Σ的命题截断　存在

“存在”不一定能”取到”, 但能”取到”则一定”存在”.

佐恩引理的表述

Zorn

对任意偏序集 U，如果 U 中所有的链都能**找到**上界，那么 U 中**存在**一个最大元.

```
Zorn = isPoset → allChainHasUb → ∃[ m ∈ U ] maximum m
```

↓ ∥∥-rec

对任意偏序集 U, 如果 U 中所有的链都**存在**上确界, 那么 U 中**存在**一个最大元

链集的链

并非直接考虑链本身, 而是考虑由所有链构成的集合在包含关系下构成的链.

现在, 假设排中律, 并给定偏序 `≤`.

```
module Chain ⦃ em : ∀ {ℓ} → EM ℓ ⦄ {U : Type u} (_≤_ : Rel U U r) where
  open import CubicalExt.Logic.Classical
  open module ≤ = Order _≤_
```

我们把 `U` 的子集 `A` 配备上链条件所得到的类型 `Chain` 叫做链集.

```
  Chain : Type _
  Chain = Σ[ A ∈ 𝒫 U ℓ-zero ] ≤.isChain A
```

可以证明 `Chain` 是集合.

```
  _ : isSet Chain
  _ = isSetΣ (isSetΠ λ _ → isSetHProp) λ _ → isProp→isSet isPropIsChain
```

链集的链(Chain)上的偏序

偏序

定义链集上的二元关系 `≤` 为链之间的包含关系.

```
_≤_ : Rel Chain Chain u
a ≤ b = a .fst ⊆ b .fst
```

由于集合的包含关系是偏序, 所以 `≤` 也是偏序.

```
≤-po : ≤.isPo
≤-po = ≤-prop , ≤-refl , ≤-antisym , ≤-trans
```

现在我们有了两个偏序, 一个是 `U` 上的 `≤`, 一个是链集上的 `≤`. 为避免混淆, 接下来我们会对相关概念加上 `≤` 或 `≤` 前缀, 来指明该概念所涉及的偏序.

链集的链(Chain)的上确界

```
sup : (A : 𝒫 Chain ℓ) → ≤.isChain A → Chain
sup A isChainA = Resize ∘ (λ x → (∃[ a@(a₁ , _ ) ∈ Chain ] x ∈ a₁ ∧ a ∈ A) , squash₁)
```

Chain 的定义仅接受 U 的位于最低宇宙的子集, 为了使我们这里定义的并确实具有 Chain 类型, 需要将上述”并”调整 (Resize) 到最低宇宙

```
suphood : (A : 𝒫 Chain ℓ) (isChainA : ≤.isChain A) → ≤.supremum A (sup A isChainA)
```

至此, 我们证明了链集中的任意链都能找到上确界.

```
≤-allChainHasSup : ≤.allChainHasSup
≤-allChainHasSup A isChainA = sup A isChainA , suphood A isChainA
```

链集的链（Chain）上的偏序 ≤ 的后继性

关键的引理

前两个前提与佐恩引理相同

第三个前提则是对佐恩引理结论的否定，将使用选择公理证明

最后将采用反证法来证明佐恩引理

```
≤-successvie : ≤.isPoset → ≤.allChainHasUb → ≤.unbound → ≤.successive
≤-successvie (Uset , ≤-po) hasUb unbnd a@(A , isChainA) = a' , resize ∘ inl , a≠a' ,
```

构造矛盾

假设排中律, 给定偏序 ≤, 假设其任意链都能取上确界, 且任意元素都取后继.

```
module Contra ⦃ em : ∀ {ℓ} → EM ℓ ⦄ {U : Type u} {_≤_ : Rel U U r}
  (≤-po : Order.isPo _≤_) (hasSup : Order.allChainHasSup _≤_) (hasSuc : Order.successiv
  open import CubicalExt.Logic.Classical
  open Order _≤_
```

归纳构造"塔"

```
data Tower : U → Type (ℓ-max (ℓ-suc ℓ-zero) (ℓ-max u r))
TowerSetℓ : 𝒫 U _
TowerSetℓ x = ∥ Tower x ∥ₚ
TowerSet : 𝒫 U ℓ-zero
TowerSet = Resize ∘ TowerSetℓ
```

归纳构造"塔"

现在归纳定义谓词 `Tower`:

- 对任意 `x` 满足 `Tower`, `x` 的后继也满足 `Tower`.
- 对任意 `U` 的子集 `A`, 如果它包含于 `TowerSetℓ`, 且是链, 那么它的上确界也满足 `Tower`.

```
data Tower where
  includeSuc : (x : U) → Tower x → Tower (hasSuc x .fst)
  includeSup : (A : 𝒫 U ℓ-zero) → (A ⊆ TowerSetℓ) → (isChainA : isChain A) →
    Tower (hasSup A isChainA .fst)
```

注意 `TowerSetℓ` 在 `Tower` 定义完成之前就被使用了. Agda 允许这种写法, 只要满足一定条件, 这里不展开.

塔也是链

```
isChainTower : ∀ x y → Tower x → Tower y → x ≤ y ∨ y ≤ x
isChainTowerSetℓ : isChain TowerSetℓ
isChainTowerSetℓ x y = rec2 squash₁ (isChainTower x y)
isChainTowerSet : isChain TowerSet
isChainTowerSet x y x∈ y∈ = isChainTowerSetℓ x y (unresize x∈) (unresize y∈)
```

```
isChainTower' : ∀ x y → Tower x → y ∈ TowerSetℓ → x ≤ y ∨ y ≤ x
isChainTower' x y x∈ | y∈ |₁ = isChainTower x y x∈ y∈
isChainTower' x y x∈ (squash₁ y∈₁ y∈₂ i) = squash₁ (isChainTower' x y x∈ y∈₁) (isChai
```

构造矛盾

证明了 TowerSet 是链之后，构造矛盾就非常简单了

由前提，TowerSet 可以取到上确界 sup，且 sup 可以取到后继 suc

按 Tower 的定义，sup 在它里面．这里命题宇宙调整（propositional resizing）起了关键作用．

```
sup∈Tower : Tower sup
sup∈Tower = includeSup TowerSet unresize isChainTowerSet
```

构造矛盾

这样，按 TowerSet 的定义，suc 也在 TowerSet 里

```
sucϵTowerSet : suc ϵ TowerSet
sucϵTowerSet = resize $ map (includeSuc sup) | supϵTower |₁
```

但是 suc 是 sup 的后继，与 sup 是 TowerSet 的上确界矛盾

```
false : ⊥
false = sup≢suc $ ≤-antisym _ _ sup≤suc suc≤sup where
  suc≤sup : suc ≤ sup
  suc≤sup = ubhood suc sucϵTowerSet
```

佐恩引理的证明

非空集合的笛卡尔积非空

$$(\forall\, x \to \| B\ x \|_1) \to \| \forall\, x \to B\ x \|_1$$

其中 x 具有类型 A, 而 A 和每个 B x 都是集合

假设选择公理, 给定 `U` 上的二元关系 `≤`.

```
module _ (ac : ∀ {ℓ ℓ'} → AC ℓ ℓ') {U : Type u} {_≤_ : Rel U U r} where
  open import CubicalExt.Logic.ClassicalChoice ac
  open Order _≤_
```

$$(\forall\, x \to \| B\ x \|_1) \to \| \forall\, x \to B\ x \|_1$$

佐恩引理的证明

```
unbound : Type _
unbound = ∀ x → Σ[ y ∈ U ] x ≤ y ∧ ¬ x ≡ y
```

目标具有适用于选择公理的形式

选择公理要求 U 是集合, 且 U 配备上无界条件也是集合, 这些都成立

```
noMaximum→unbound : isPoset → ¬ (∃[ m ∈ U ] maximum m) → ∥ unbound ∥₁
noMaximum→unbound ≤-poset noMax = ac Uset Σset H where
  Uset = ≤-poset .fst
  ≤-prop = ≤-poset .snd .fst
  Σset : ∀ x → isSet (Σ[ x' ∈ U ] (x ≤ x' ∧ ¬ x ≡ x'))
  Σset = λ _ → isSetΣ Uset λ _ → isProp→isSet $ isPropΣ (≤-prop _ _) λ _ → isPropΠ λ
```

```
  H₀ : ∀ x → ∃[ x' ∈ U ] ¬ (x ≤ x' → x ≡ x')
  H₀ x = ¬∀→∃¬ λ H → noMax ∣ x , H ∣₁
  H : ∀ x → ∃[ x' ∈ U ] (x ≤ x' ∧ ¬ x ≡ x')
  H x = rec squash₁ (λ { (x' , H) → ∣ x' , ¬→→∧ (x ≤ x') (x ≡ x') H ∣₁ }) (H₀ x)
```

佐恩引理的证明

$$(\forall\, x \to \| B\ x \|_1) \to \| \forall\, x \to B\ x \|_1$$

```
unbound : Type _
unbound = ∀ x → Σ[ y ∈ U ] x ≤ y ∧ ¬ x ≡ y
```

```
noMaximum→unbound : isPoset → ¬ (∃[ m ∈ U ] maximum m) → ∥ unbound ∥₁
noMaximum→unbound ≤-poset noMax = ac Uset Σset H where
```

```
    H₀ : ∀ x → ∃[ x' ∈ U ] ¬ (x ≤ x' → x ≡ x')
    H₀ x = ¬∀→∃¬ λ H → noMax ∣ x , H ∣₁
    H : ∀ x → ∃[ x' ∈ U ] (x ≤ x' ∧ ¬ x ≡ x')
    H x = rec squash₁ (λ { (x' , H) → ∣ x' , ¬→→∧ (x ≤ x') (x ≡ x') H ∣₁ }) (H₀ x)
```

佐恩引理的证明

最后佐恩引理的证明就非常简单了. 用反证法, 假设 `U` 没有最大元, 由上一条引理有 `// unbound //₁`. 这时只需 `rec` 到 `⊥`, 所以可以去掉截断, 拿到完整的 `unbound`. 这正好是 `≤-successvie` 的前提, 于是我们可以证明链集的任意链都能取上界且链集是后继的. 由"塔"的构造我们知道这是矛盾的.

```
zorn : Zorn
zorn ≤-poset hasUb = byContra λ noMax → rec isProp⊥
  (Contra.false ≤-po ≤-allChainHasSup ∘ ≤-successvie ≤-poset hasUb)
  (noMaximum→unbound ≤-poset noMax)
  where open Chain _≤_
```