Lean4 を用いた Gödel の第一/第二不完全性定理の形式化

齋藤彰悟

東北大学大学院理学研究科数学専攻

2024-12-05

0.1 FFL

形式論理学の形式化プロジェクト： `github.com/FormalizedFormalLogic` 

- 古典命題論理 `LO.Propositional`
- 直観主義命題論理 `LO.IntProp`
- 古典命題様相論理 `LO.Modal`
- **古典一階述語論理 `LO.FirstOrder`**
- 直観主義一階述語論理 `LO.IntFO`

0.2 不完全性定理

理論 T を計算可能な一階算術の理論とする.

定理 0.2.1 (不完全性定理):
(G1) T が基礎的な算術を扱える程度に強く，まともならば，T から証明も反証もできない論理式が存在する.
(G2) T が十分強く，無矛盾ならば，T の無矛盾性を表す文は証明できない.

1986(!), Shanker Nqthm による G1 の形式化.
2004, O'Connor Coq による G1 の形式化.
2004, Harrison HOL Light による G1 の形式化.

0.2 不完全性定理

2021, Paulson Isabelle/HOL による G1 と G2 の形式化 [1]. 算術ではなく遺伝的有限集合の理論 HF (ペアノ算術 PA と同等)の上で証明している.

- 私が知る限り唯一の第二不完全性定理の形式化[1].

定理 0.2.2 (Formalized by Paulson):
(G1) HF からは証明も反証もできない論理式が存在する.
(G2) 自己の無矛盾性を表す文は HF からは証明できない.

[1]ただし可証性条件を仮定した上での G2 の証明はいくつか存在する.

0.2 不完全性定理

次の不完全性定理の一つのバリエーションを形式化した[1]：

定理 0.2.3:
(G1) T が R_0 より強く Σ_1-健全ならば，T から証明も反証もできない論理式が存在する.
(G2) T が $\mathsf{I}\Sigma_1$ より強く無矛盾ならば，T の無矛盾性を表す文は証明できない.

- Σ_1-健全：T から証明可能な Σ_1 文は標準モデルの上で真.
- R_0 : Cobham の最弱の算術.
- $\mathsf{I}\Sigma_1$: ペアノ算術の断片理論.

[1]後述するように完全性定理を用いているため非構成的.

0.2 不完全性定理

以下の 5 つのステップを踏んで証明を行う.

1. 一階述語論理の形式化.
    - 項と論理式，代入操作などの形式化.
    - 証明可能性，充足性の形式化.
2. 完全性定理.
    - 証明探索
3. $\mathrm{I}\Sigma_1$ の内部で算術を展開する.
    - 指数関数が定義可能であることを証明する.
    - 有限集合，有限列といった基礎概念をコード化する.
4. メタ数学の算術化.
    - 項，論理式，証明可能性などをコード化する.
5. Hilbert-Bernays-Löb の可証性条件

1. 一階述語論理の形式化.

1.1 項・論理式の形式化

論理式（擬論理式）は常に否定標準形を取るように定義する[1]. すなわち,

$$\varphi, \psi ::= \top \mid \bot \mid R(\vec{v}) \mid \neg R(\vec{v}) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \forall \varphi \mid \exists \varphi$$

```
inductive Semiformula (L : Language) (ξ : Type*) : ℕ → Type _ where
  | verum  {n} : Semiformula L ξ n
  | falsum {n} : Semiformula L ξ n
  | rel    {n} : {arity : ℕ} → L.Rel arity → (Fin arity → Semiterm L ξ n) → Semiformula L ξ n
  | nrel   {n} : {arity : ℕ} → L.Rel arity → (Fin arity → Semiterm L ξ n) → Semiformula L ξ n
  | and    {n} : Semiformula L ξ n → Semiformula L ξ n → Semiformula L ξ n
  | or     {n} : Semiformula L ξ n → Semiformula L ξ n → Semiformula L ξ n
  | all    {n} : Semiformula L ξ (n + 1) → Semiformula L ξ n
  | ex     {n} : Semiformula L ξ (n + 1) → Semiformula L ξ n
```

論理式	糖衣構文
$\overline{98} + \overline{6748} = \overline{6846}$	“98 + 6748 = 6846”
$(\forall x)(\forall y)[a \cdot (x + y) = a \cdot x + a \cdot y]$	“a ∣ ∀ x y, a * (x + y) = a * x + a * y”
$(\forall x)\left[x < \overline{n} \leftrightarrow \bigvee_{i<n} x = \vec{i}\right]$	“∀ x, x < ↑n ↔ ∨ i < n, x = ↑i”

[1]否定 $\neg$ は論理式の関数として定義する. また $\varphi \to \psi := \neg\varphi \vee \psi$ とする.

1.2 証明可能性の形式化

証明体系には Tait 計算を採用する.

$$\frac{}{R(\vec{v}), \neg R(\vec{v}), \Delta}\,\mathtt{axL} \qquad \frac{}{\top, \Delta}\,\mathtt{verum} \qquad \frac{\varphi, \psi, \Delta}{\varphi \vee \psi, \Delta}\,\mathtt{or} \qquad \frac{\varphi, \Delta \quad \psi, \Delta}{\varphi \wedge \psi, \Delta}\,\mathtt{and} \qquad \frac{\varphi^+(\&0), \Delta^+}{\forall \varphi, \Delta}\,\mathtt{all}^1 \qquad \frac{\varphi(t), \Delta}{\exists \varphi, \Delta}\,\mathtt{ex}$$

$$\frac{\Delta}{\Gamma}\,\mathtt{wk} \quad (\text{if } \Delta \subseteq \Gamma) \qquad \frac{\varphi, \Delta \quad \neg\varphi, \Delta}{\Delta}\,\mathtt{cut} \qquad \frac{}{\varphi}\,\mathtt{root} \quad (\text{if } \varphi \in T)$$

```
inductive Derivation (T : Theory L) : Sequent L → Type _
  | axL (Δ) {k} (r : L.Rel k) (v) : Derivation T (rel r v :: nrel r v :: Δ)
  | verum (Δ)    : Derivation T (⊤ :: Δ)
  | or {Δ p q}   : Derivation T (p :: q :: Δ) → Derivation T (p ⋎ q :: Δ)
  | and {Δ p q}  : Derivation T (p :: Δ) → Derivation T (q :: Δ) → Derivation T (p ⋏ q :: Δ)
  | all {Δ p}    : Derivation T (Rew.free.hom p :: Δ⁺) → Derivation T ((∀' p) :: Δ)
  | ex {Δ p} (t) : Derivation T (p/[t] :: Δ) → Derivation T ((∃' p) :: Δ)
  | wk {Δ Γ}     : Derivation T Δ → Δ ⊆ Γ → Derivation T Γ
  | cut {Δ p}    : Derivation T (p :: Δ) → Derivation T (~p :: Δ) → Derivation T Δ
  | root {p}     : p ∈ T → Derivation T [p]
```

Tait 計算は推論規則が少ないため扱いやすく，LK などの他の推論規則に比べ様々な証明が簡易になる（こともある）.

[1] ここでは自由変数を $\&0, \&1, \ldots$ と表記する．また φ^+, Γ^+ はそれぞれに含まれる自由変数をインクリメントしたもの

2. 完全性定理

2.1 完全性定理

定理 2.1.1 (完全性定理): すべての 論理式 φ について,

$$T \vdash \varphi \Longleftrightarrow T \vDash \varphi$$

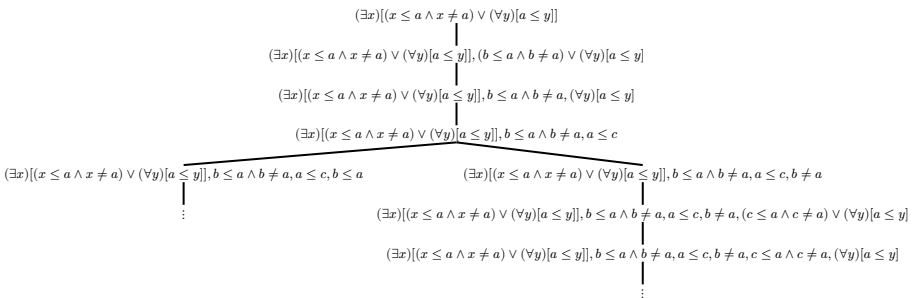
$\Longrightarrow$ 方向（健全性定理）は証明に関する帰納法により従う．$\Longleftarrow$ を示すには以下を証明すれば良い.

補題 2.1.1: 推件 Γ について， 次のいずれかが成立する.

- $T \vdash_{\mathsf{T}} \Gamma$
- すべての $\varphi \in \Gamma$ を充足しないような T のモデルが存在する.

2.2 証明探索

証明は Tait 計算の証明探索による(参考: [2]).




```
inductive Redux (T : Theory L) : Code L → Sequent L → Sequent L → Prop
  | axLRefl   {Γ : Sequent L} {k} (τ : L.Rel k) (v) :
    Semiformula.rel τ v ∉ Γ ∨ Semiformula.nrel τ v ∉ Γ → Redux T (Code.axL τ v) Γ Γ
  | verumRefl {Γ : Sequent L} : ⊤ ∉ Γ → Redux T Code.verum Γ Γ
  | and₁      {Γ : Sequent L} {p q : SyntacticFormula L} : p ⋏ q ∈ Γ → Redux T (Code.and p q) (p :: Γ) Γ
  | and₂      {Γ : Sequent L} {p q : SyntacticFormula L} : p ⋏ q ∈ Γ → Redux T (Code.and p q) (q :: Γ) Γ
  | andRefl   {Γ : Sequent L} {p q : SyntacticFormula L} : p ⋏ q ∉ Γ → Redux T (Code.and p q) Γ Γ
  | or        {Γ : Sequent L} {p q : SyntacticFormula L} : p ⋎ q ∈ Γ → Redux T (Code.or p q) (p :: q :: Γ) Γ
  | orRefl    {Γ : Sequent L} {p q : SyntacticFormula L} : p ⋎ q ∉ Γ → Redux T (Code.or p q) Γ Γ
  | all       {Γ : Sequent L} {p : SyntacticSemiformula L 1} : ∀' p ∈ Γ → Redux T (Code.all p) (p/[&(newVar Γ)] :: Γ) Γ
  | allRefl   {Γ : Sequent L} {p : SyntacticSemiformula L 1} : ∀' p ∉ Γ → Redux T (Code.all p) Γ Γ
  | ex        {Γ : Sequent L} {p : SyntacticSemiformula L 1} {t : SyntacticTerm L} :
    ∃' p ∈ Γ → Redux T (Code.ex p t) (p/[t] :: Γ) Γ
  | exRefl    {Γ : Sequent L} {p : SyntacticSemiformula L 1} {t : SyntacticTerm L} :
    ∃' p ∉ Γ → Redux T (Code.ex p t) Γ Γ
  | id        {Γ : Sequent L} {p : SyntacticFormula L} (hp : p ∈ T) : Redux T (Code.id p) ((~∀∀p) :: Γ) Γ
  | idRefl    {Γ : Sequent L} {p : SyntacticFormula L} (hp : p ∉ T) : Redux T (Code.id p) Γ Γ

local notation:25 Δ₁" ≺[" c:25 "] " Δ₂:80 ⇒ Redux T c Δ₁ Δ₂

inductive ReduxNat (T : Theory L) (s : ℕ) : Sequent L → Sequent L → Prop
  | redux {c : Code L} : decode s.unpair.1 = some c → ∀ {Δ₂ Δ₁}, Redux T c Δ₂ Δ₁ → ReduxNat T s Δ₂ Δ₁
  | refl : decode (α := Code L) s.unpair.1 = none → ∀ Δ, ReduxNat T s Δ Δ

local notation:25 Δ₁" ≺⟨" s:25 "⟩ " Δ₂:80 ⇒ ReduxNat T s Δ₁ Δ₂
```

探索木が well-founded ならば証明可能.

```
noncomputable def syntacticMainLemma
    (wf : WellFounded (SearchTree.Lt T Γ))
    (p : SearchTree T Γ) :
    T ⇒ p.seq
```

探索木が well-founded でないならば Γ の反例モデル `Model T Γ` が構成できる.

```
lemma Model.models : Model T Γ ⊨ₘ* T

lemma semanticMainLemmaTop
    (nwf : ¬WellFounded (SearchTree.Lt T Γ))
    {φ : SyntacticFormula L}
    (h : φ ∈ Γ) :
    ¬Evalf (Model.structure T Γ) Semiterm.fvar φ
```

3. $\mathrm{I}\Sigma_1$の内部で算術を展開する

3.1 算術の公理系を定義する

PA^-

```
inductive PeanoMinus : Theory ℒₒᵣ
  | equal          : ∀ φ ∈ EQ, PeanoMinus φ
  | addZero        : PeanoMinus “x | x + 0 = x”
  | addAssoc       : PeanoMinus “x y z | (x + y) + z = x + (y + z)”
  | addComm        : PeanoMinus “x y | x + y = y + x”
  | addEqOfLt      : PeanoMinus “x y | x < y → ∃ z, x + z = y”
  | zeroLe         : PeanoMinus “x | 0 ≤ x”
  | zeroLtOne      : PeanoMinus “0 < 1”
  | oneLeOfZeroLt  : PeanoMinus “x | 0 < x → 1 ≤ x”
  | addLtAdd       : PeanoMinus “x y z | x < y → x + z < y + z”
  | mulZero        : PeanoMinus “x | x * 0 = 0”
  | mulOne         : PeanoMinus “x | x * 1 = x”
  | mulAssoc       : PeanoMinus “x y z | (x * y) * z = x * (y * z)”
  | mulComm        : PeanoMinus “x y | x * y = y * x”
  | mulLtMul       : PeanoMinus “x y z | x < y ∧ 0 < z → x * z < y * z”
  | distr          : PeanoMinus “x y z | x * (y + z) = x * y + x * z”
  | ltIrrefl       : PeanoMinus “x | x ≮ x”
  | ltTrans        : PeanoMinus “x y z | x < y ∧ y < z → x < z”
  | ltTri          : PeanoMinus “x y | x < y ∨ x = y ∨ x > y”
```

3.1 算術の公理系を定義する

$I\varphi$

```
def succInd {ξ} (φ : Semiformula L ξ 1) : Formula L ξ :=
  “!φ 0 → (∀ x, !φ x → !φ (x + 1)) → ∀ x, !φ x”

def indScheme (Γ : Semiformula L ℕ 1 → Prop) : Theory L :=
  { q | ∃ p : Semiformula L ℕ 1, Γ p ∧ q = succInd p }
```

$I\Sigma_k$

```
abbrev indH (Γ : Polarity) (k : ℕ) : Theory ℒₒᵣ :=
  𝐏𝐀⁻ + indScheme ℒₒᵣ (Arith.Hierarchy Γ k)

abbrev iSigma (k : ℕ) : Theory ℒₒᵣ := 𝐈𝐍𝐃 𝚺 k
```

PA

```
abbrev peano : Theory ℒₒᵣ := 𝐏𝐀⁻ + indScheme ℒₒᵣ Set.univ
```

R_0

```
inductive CobhamR0 : Theory ℒₒᵣ
  | equal         : ∀ φ ∈ 𝐄𝐐, CobhamR0 φ
  | Ω₁ (n m : ℕ) : CobhamR0 “↑n + ↑m = ↑(n + m)”
  | Ω₂ (n m : ℕ) : CobhamR0 “↑n * ↑m = ↑(n * m)”
  | Ω₃ (n m : ℕ) : n ≠ m → CobhamR0 “↑n ≠ ↑m”
  | Ω₄ (n : ℕ)   : CobhamR0 “∀ x, x < ↑n ↔ ∨ i < n, x = ↑i”
```

3.2 （形式的）証明のルーチン

何らかの算術の体系 T からある論理式を証明できること を証明したい.

$$\mathsf{Lean} \vdash \text{“}T \vdash \varphi\text{”}$$

しかし，そもそも Lean が形式化された数学であり，その内部でさらに形式化された証明を証明するのはかなり煩雑になる （さらに後には「形式化された形式化された形式化された証明」のようなものさえ扱うことになる）.

これは大変なので， 完全性定理を用いて代わりに意味論帰結を証明する.

$$\mathsf{Lean} \vdash \text{“}T \vDash \varphi\text{”}$$

意味論的な議論では，Lean のライブラリに用意された代数学の種々の補題やメタプログラミングや自動証明が利用できるため，より簡単に作業を行うことができる.

3.2 （形式的）証明のルーチン

まず理論の任意のモデル V を固定する.

```
variable {V : Type*} [ORingStruc V] [V ⊨ₘ* Iopen]
```

- **ORingStruc V:** V が言語 $\mathcal{L}_{\mathrm{OR}}$ の構造であることを主張する typeclass.
- **V ⊨ₘ* Iopen:** V が理論 Iopen を満たすことを主張する typeclass.

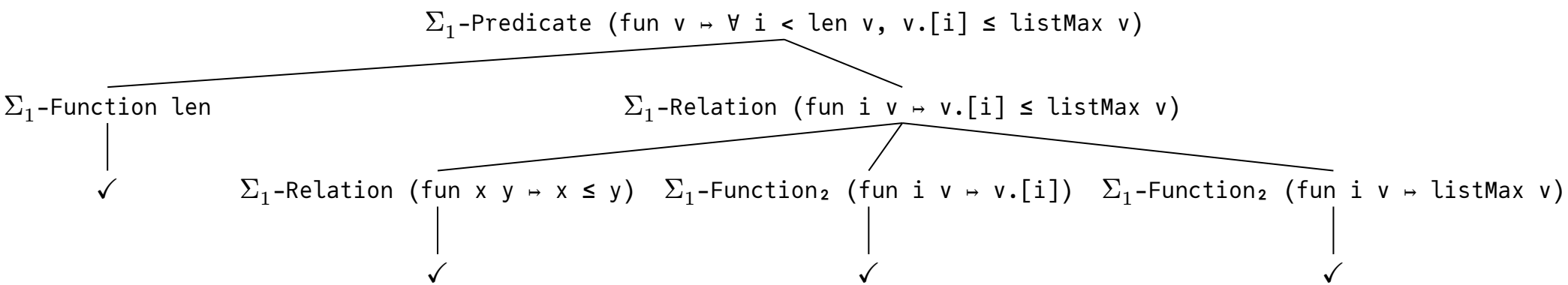
この仮定のもとで証明を行う． 関数を追加したいならば選択関数を用いる.

```
lemma sqrt_exists_unique (a : V) :
    ∃! x, x * x ≤ a ∧ a < (x + 1) * (x + 1) := by ...
def sqrt (a : V) : V := Classical.choose! (sqrt_exists_unique a)
prefix:75 "√" ⇒ sqrt
@[simp] lemma sqrt_mul_self (a : V) : √(a * a) = a := by ...
```

ただし，帰納法を適用するために，定義した関数や関係を含む述語がある算術的階層に含まれることが示せなければならない.

3.3 definability tactic

definability tactic は Lean の述語がある算術的階層に含まれることを（可能なら）自動証明する.



typeclass に登録された関数や関係は自動証明の末尾を示すために使用される.

```
instance exp_definable : Σ₀-Function₁ (Exp.exp : V → V) := by ...
instance length_definable : Σ₀-Function₁ (‖·‖ : V → V) := by ...
instance dvd_definable : Σ₀-Relation (fun a b : V ↦ a ∣ b) := by ...
instance Language.isSemiterm_definable : Δ₁-Relation L.IsSemiterm := by ...
```

3.4 指数関数

指数関数のグラフ $\mathrm{Exp}(x, y) \Leftrightarrow 2^x = y$ は Σ_0 論理式で定義可能 [3], [4].

$$
\mathrm{Exp}(a_0, a_1) \equiv [[\mathrm{a}_0 = 0] \land [a_1 = 1]] \lor [(\exists x_0 < ((((a_1 \cdot a_1) \cdot a_1) \cdot a_1) + 1))[(\exists x_1 < ((((a_1 \cdot a_1) \cdot a_1) \cdot a_1) + 1))[[[(\exists x_2 < (x_0 + 1))[[[[0 \not< (((1 + 1) + 1) + 1)] \lor [(\exists x_3 < (((1 + 1) + 1) + 1))[x_0 =
(((((1+1)+1)+1)\cdot x_2)+x_3)]]] \land [[((((1+1)+1)+1) \neq 0] \lor [x_2 = 0]]] \land [(\exists x_3 < (x_2+1))[[[[0 \not< (((1+1)+1)+1)] \lor [(\exists x_4 < (((1+1)+1)+1))[x_2 = (((((1+1)+1)+1)\cdot x_3)+x_4)]]] \land [[((((1+1)+1)+1) \neq 0] \lor [x_3 =
0]]] \land [[[[(((((1+1)+1)+1)\cdot x_3) \neq x_2] \land [((((1+1)+1)+1)\cdot x_3) \not< x_2]] \lor [x_2 = (((((1+1)+1)+1)\cdot x_3)+1)]] \land [[x_2 \not< ((((1+1)+1)+1)\cdot x_3)] \lor [1 = 0]]]]]]] \land [(\exists x_2 < (x_1+1))[[[[0 \not< (((1+1)+1)+1)] \lor [(\exists x_3 <
(((1+1)+1)+1))[x_1 = (((((1+1)+1)+1)\cdot x_2)+x_3)]]] \land [[((((1+1)+1)+1) \neq 0] \lor [x_2 = 0]]] \land [(\exists x_3 < (x_2+1))[[[[0 \not< (((1+1)+1)+1)] \lor [(\exists x_4 < (((1+1)+1)+1))[x_2 = (((((1+1)+1)+1)\cdot x_3)+x_4)]]] \land [[((((1+1)+1)+1) \neq
0] \lor [x_3 = 0]]] \land [[[[(((((1+1)+1)+1)\cdot x_3) \neq x_2] \land [((((1+1)+1)+1)\cdot x_3) \not< x_2]] \lor [x_2 = (((((1+1)+1)+1)\cdot x_3)+(1+1))]] \land [[x_2 \not< ((((1+1)+1)+1)\cdot x_3)] \lor [(1+1) = 0]]]]]]]] \land [[(\forall x_2 < (a_1+1))[[x_2 = (1+1)] \lor [[[[0 \not<
x_2] \lor [(\exists x_3 < (x_2+1))[[1 < x_3] \land [[(\exists x_4 < (x_2+1))[x_2 = (x_3 \cdot x_4)]] \land [(\forall x_4 < (x_3+1))[x_3 \neq ((1+1)\cdot x_4)]]]]]] \lor [(\forall x_3 < ((1+1)\cdot x_2))[[[(\exists x_4 < (x_3+1))[[[[0 \not< 1] \lor [(\exists x_5 < 1)[x_3 = ((1\cdot x_4)+x_5)]]] \land [[1 \neq 0] \lor [x_4 =
0]]] \land [(\forall x_5 < (x_4+1))[x_4 \neq ((1+1)\cdot x_5)]]]] \lor [[(\forall x_4 < (x_3+1))[[[[0 < (1+1)] \land [(\forall x_5 < (1+1))[x_3 \neq (((1+1)\cdot x_4)+x_5)]]] \lor [[(1+1) = 0] \land [x_4 \neq 0]]] \lor [(\exists x_5 < (x_4+1))[x_4 = ((1+1)\cdot x_5)]]]] \lor [(\exists x_4 < (x_3+1))[[[0 <
x_4] \land [(\forall x_5 < (x_4+1))[[1 \not< x_5] \lor [[(\forall x_6 < (x_4+1))[x_4 \neq (x_5 \cdot x_6)]] \lor [(\exists x_6 < (x_5+1))[x_5 = ((1+1)\cdot x_6)]]]]]] \land [[(1+1) < x_4] \land [[[(\exists x_5 < (x_3+1))[[[[0 \not< x_4] \lor [(\exists x_6 < x_4)[x_3 = ((x_4 \cdot x_5)+x_6)]]] \land [[x_4 \neq 0] \lor [x_5 =
0]]] \land [(\forall x_6 < (x_5+1))[x_5 \neq ((1+1)\cdot x_6)]]]] \land [(\forall x_5 < (x_4+1))[[[[(x_5\cdot x_5) \neq x_4] \land [(x_5\cdot x_5) \not< x_4]] \lor [x_4 \not< ((x_5+1)\cdot(x_5+1))]] \lor [[(x_5\cdot x_5) \neq x_4] \lor [(\forall x_6 < (x_3+1))[[[[0 < x_5] \land [(\forall x_7 < x_5)[x_3 \neq ((x_5\cdot x_6)+x_7)]]] \lor [[x_5 =
0] \land [x_6 \neq 0]]] \lor [(\exists x_7 < (x_6+1))[x_6 = ((1+1)\cdot x_7)]]]]]]]] \lor [[(\exists x_5 < (x_4+1))[[[[(x_5\cdot x_5) = x_4] \lor [(x_5\cdot x_5) < x_4]] \land [x_4 < ((x_5+1)\cdot(x_5+1))]] \land [[(x_5\cdot x_5) = x_4] \land [(\exists x_6 < (x_3+1))[[[[0 \not< x_5] \lor [(\exists x_7 < x_5)[x_3 =
((x_5\cdot x_6)+x_7)]]] \land [[x_5 \neq 0] \lor [x_6 = 0]]] \land [(\forall x_7 < (x_6+1))[x_6 \neq ((1+1)\cdot x_7)]]]]]]] \land [(\forall x_5 < (x_3+1))[[[[0 < x_4] \land [(\forall x_6 < x_4)[x_3 \neq ((x_4\cdot x_5)+x_6)]]] \lor [[x_4 = 0] \land [x_5 \neq 0]]] \lor [(\exists x_6 < (x_5+1))[x_5 = ((1+1)\cdot x_6)]]]]]]]]]] \lor [(\forall x_4 <
(x_3+1))[[[[0 < x_2] \land [(\forall x_5 < x_2)[x_3 \neq ((x_2\cdot x_4)+x_5)]]] \lor [[x_2 = 0] \land [x_4 \neq 0]]] \lor [(\exists x_5 < (x_4+1))[x_4 = ((1+1)\cdot x_5)]]]]]]] \lor [[[(\exists x_3 < (x_0+1))[[(\exists x_4 < (x_0+1))[[[[0 \not< x_2] \lor [(\exists x_5 < x_2)[x_0 = ((x_2\cdot x_4)+x_5)]]] \land [[x_2 \neq
0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2\cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2\cdot x_5) \neq x_4] \land [(x_2\cdot x_5) \not< x_4]] \lor [x_4 = ((x_2\cdot x_5)+x_3)]] \land [[x_4 \not< (x_2\cdot x_5)] \lor [x_3 = 0]]]]]]] \land [(\exists x_4 < (x_0+1))[[[[0 \not<
(x_2\cdot x_2)] \lor [(\exists x_5 < (x_2\cdot x_2))[x_0 = (((x_2\cdot x_2)\cdot x_4)+x_5)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< (x_2\cdot x_2)] \lor [(\exists x_6 < (x_2\cdot x_2))[x_4 = (((x_2\cdot x_2)\cdot x_5)+x_6)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2\cdot x_2)\cdot x_5) \neq
x_4] \land [((x_2\cdot x_2)\cdot x_5) \not< x_4]] \lor [x_4 = (((x_2\cdot x_2)\cdot x_5)+((1+1)\cdot x_3))]] \land [[x_4 \not< ((x_2\cdot x_2)\cdot x_5)] \lor [((1+1)\cdot x_3) = 0]]]]]]]] \land [(\exists x_3 < (x_1+1))[[(\exists x_4 < (x_1+1))[[[[0 \not< x_2] \lor [(\exists x_5 < x_2)[x_1 = ((x_2\cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 =
0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2\cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2\cdot x_5) \neq x_4] \land [(x_2\cdot x_5) \not< x_4]] \lor [x_4 = ((x_2\cdot x_5)+x_3)]] \land [[x_4 \not< (x_2\cdot x_5)] \lor [x_3 = 0]]]]]]] \land [(\exists x_4 < (x_1+1))[[[[0 \not<
(x_2\cdot x_2)] \lor [(\exists x_5 < (x_2\cdot x_2))[x_1 = (((x_2\cdot x_2)\cdot x_4)+x_5)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< (x_2\cdot x_2)] \lor [(\exists x_6 < (x_2\cdot x_2))[x_4 = (((x_2\cdot x_2)\cdot x_5)+x_6)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2\cdot x_2)\cdot x_5) \neq
x_4] \land [((x_2\cdot x_2)\cdot x_5) \not< x_4]] \lor [x_4 = (((x_2\cdot x_2)\cdot x_5)+(x_3\cdot x_3))]] \land [[x_4 \not< ((x_2\cdot x_2)\cdot x_5)] \lor [(x_3\cdot x_3) = 0]]]]]]]]]] \lor [[(\exists x_3 < (x_0+1))[[(\exists x_4 < (x_0+1))[[[[0 \not< x_2] \lor [(\exists x_5 < x_2)[x_0 = ((x_2\cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 =
0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2\cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2\cdot x_5) \neq x_4] \land [(x_2\cdot x_5) \not< x_4]] \lor [x_4 = ((x_2\cdot x_5)+x_3)]] \land [[x_4 \not< (x_2\cdot x_5)] \lor [x_3 = 0]]]]]]] \land [(\exists x_4 < (x_0+1))[[[[0 \not<
(x_2\cdot x_2)] \lor [(\exists x_5 < (x_2\cdot x_2))[x_0 = (((x_2\cdot x_2)\cdot x_4)+x_5)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< (x_2\cdot x_2)] \lor [(\exists x_6 < (x_2\cdot x_2))[x_4 = (((x_2\cdot x_2)\cdot x_5)+x_6)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2\cdot x_2)\cdot x_5) \neq
x_4] \land [((x_2\cdot x_2)\cdot x_5) \not< x_4]] \lor [x_4 = (((x_2\cdot x_2)\cdot x_5)+(((1+1)\cdot x_3)+1))]] \land [[x_4 \not< ((x_2\cdot x_2)\cdot x_5)] \lor [(((1+1)\cdot x_3)+1) = 0]]]]]]]] \land [(\exists x_3 < (x_1+1))[[(\exists x_4 < (x_1+1))[[[[0 \not< x_2] \lor [(\exists x_5 < x_2)[x_1 = ((x_2\cdot x_4)+x_5)]]] \land [[x_2 \neq
0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2\cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2\cdot x_5) \neq x_4] \land [(x_2\cdot x_5) \not< x_4]] \lor [x_4 = ((x_2\cdot x_5)+x_3)]] \land [[x_4 \not< (x_2\cdot x_5)] \lor [x_3 = 0]]]]]]] \land [(\exists x_4 < (x_1+1))[[[[0 \not<
(x_2\cdot x_2)] \lor [(\exists x_5 < (x_2\cdot x_2))[x_1 = (((x_2\cdot x_2)\cdot x_4)+x_5)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \not< (x_2\cdot x_2)] \lor [(\exists x_6 < (x_2\cdot x_2))[x_4 = (((x_2\cdot x_2)\cdot x_5)+x_6)]]] \land [[(x_2\cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2\cdot x_2)\cdot x_5) \neq
x_4] \land [((x_2\cdot x_2)\cdot x_5) \not< x_4]] \lor [x_4 = (((x_2\cdot x_2)\cdot x_5)+((1+1)\cdot(x_3\cdot x_3)))]] \land [[x_4 \not< ((x_2\cdot x_2)\cdot x_5)] \lor [((1+1)\cdot(x_3\cdot x_3)) = 0]]]]]]]]]]]]]] \land [(\exists x_2 < ((a_1\cdot a_1)+1))[[x_2 \neq (1+1)] \land [[[[0 < x_2] \land [(\forall x_3 < (x_2+1))[[1 \not<
x_3] \lor [[(\forall x_4 < (x_2+1))[x_2 \neq (x_3\cdot x_4)]] \lor [(\exists x_4 < (x_3+1))[x_3 = ((1+1)\cdot x_4)]]]]]] \land [(\exists x_3 < ((1+1)\cdot x_2))[[[(\forall x_4 < (x_3+1))[[[[0 < 1] \land [(\forall x_5 < 1)[x_3 \neq ((1\cdot x_4)+x_5)]]] \lor [[1 = 0] \land [x_4 \neq 0]]] \lor [(\exists x_5 < (x_4+1))[x_4 =
((1+1)\cdot x_5)]]]] \land [[(\exists x_4 < (x_3+1))[[[[0 \not< (1+1)] \lor [(\exists x_5 < (1+1))[x_3 = (((1+1)\cdot x_4)+x_5)]]] \land [[(1+1) \neq 0] \lor [x_4 = 0]]] \land [(\forall x_5 < (x_4+1))[x_4 \neq ((1+1)\cdot x_5)]]]] \land [(\forall x_4 < (x_3+1))[[[0 \not< x_4] \lor [(\exists x_5 < (x_4+1))[[1 <
x_5] \land [[(\exists x_6 < (x_4+1))[x_4 = (x_5\cdot x_6)]] \land [(\forall x_6 < (x_5+1))[x_5 \neq ((1+1)\cdot x_6)]]]]]] \lor [[(1+1) \not< x_4] \lor [[[(\forall x_5 < (x_3+1))[[[[0 < x_4] \land [(\forall x_6 < x_4)[x_3 \neq ((x_4\cdot x_5)+x_6)]]] \lor [[x_4 = 0] \land [x_5 \neq 0]]] \lor [(\exists x_6 < (x_5+1))[x_5 =
((1+1)\cdot x_6)]]]] \lor [(\exists x_5 < (x_4+1))[[[[(x_5\cdot x_5) = x_4] \lor [(x_5\cdot x_5) < x_4]] \land [x_4 < ((x_5+1)\cdot(x_5+1))]] \land [[(x_5\cdot x_5) = x_4] \land [(\exists x_6 < (x_3+1))[[[[0 \not< x_5] \lor [(\exists x_7 < x_5)[x_3 = ((x_5\cdot x_6)+x_7)]]] \land [[x_5 \neq 0] \lor [x_6 = 0]]] \land [(\forall x_7 <
(x_6+1))[x_6 \neq ((1+1)\cdot x_7)]]]]]]]] \land [[(\forall x_5 < (x_4+1))[[[[(x_5\cdot x_5) \neq x_4] \land [(x_5\cdot x_5) \not< x_4]] \lor [x_4 \not< ((x_5+1)\cdot(x_5+1))]] \lor [[(x_5\cdot x_5) \neq x_4] \lor [(\forall x_6 < (x_3+1))[[[[0 < x_5] \land [(\forall x_7 < x_5)[x_3 \neq ((x_5\cdot x_6)+x_7)]]] \lor [[x_5 = 0] \land [x_6 \neq
0]]] \lor [(\exists x_7 < (x_6+1))[x_6 = ((1+1)\cdot x_7)]]]]]]] \lor [(\exists x_5 < (x_3+1))[[[[0 \not< x_4] \lor [(\exists x_6 < x_4)[x_3 = ((x_4\cdot x_5)+x_6)]]] \land [[x_4 \neq 0] \lor [x_5 = 0]]] \land [(\forall x_6 < (x_5+1))[x_5 \neq ((1+1)\cdot x_6)]]]]]]]]]]] \land [(\exists x_4 < (x_3+1))[[[[0 \not< x_2] \lor [(\exists x_5 <
x_2)[x_3 = ((x_2\cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [(\forall x_5 < (x_4+1))[x_4 \neq ((1+1)\cdot x_5)]]]]]]] \land [[(\exists x_3 < (x_0+1))[[[[0 \not< x_2] \lor [(\exists x_4 < x_2)[x_0 = ((x_2\cdot x_3)+x_4)]]] \land [[x_2 \neq 0] \lor [x_3 = 0]]] \land [(\exists x_4 < (x_3+1))[[[[0 \not< x_2] \lor [(\exists x_5 <
x_2)[x_3 = ((x_2\cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [[[[(x_2\cdot x_4) \neq x_3] \land [(x_2\cdot x_4) \not< x_3]] \lor [x_3 = ((x_2\cdot x_4)+a_0)]] \land [[x_3 \not< (x_2\cdot x_4)] \lor [a_0 = 0]]]]]]] \land [(\exists x_3 < (x_1+1))[[[[0 \not< x_2] \lor [(\exists x_4 < x_2)[x_1 = ((x_2\cdot x_3)+x_4)]]] \land [[x_2 \neq
0] \lor [x_3 = 0]]] \land [(\exists x_4 < (x_3+1))[[[[0 \not< x_2] \lor [(\exists x_5 < x_2)[x_3 = ((x_2\cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [[[[(x_2\cdot x_4) \neq x_3] \land [(x_2\cdot x_4) \not< x_3]] \lor [x_3 = ((x_2\cdot x_4)+a_1)]] \land [[x_3 \not< (x_2\cdot x_4)] \lor [a_1 = 0]]]]]]]]]]]]]]]]
$$

3.5 遺伝的有限集合

「x は y の要素」 $\iff$ 「y を二進数展開したとき x 桁目が 1」 と定義する.

$$x \in y \iff \mathrm{Bit}(x, y) \iff \lfloor y/2^x \rfloor \bmod 2 = 1$$

Ackermann coding によって遺伝的有限集合 V_ω が扱える. $I\Sigma_1$ のもとで基礎的な集合論が展開できる.

```
theorem finset_comprehension₁ {Γ} {P : V → Prop} (hP : Γ-[1]-Predicate P) (a : V) :
    ∃ s < exp a, ∀ i < a, i ∈ s ↔ P i -- (制限された)内包公理図式
theorem sUnion_exists_unique (s : V) :
    ∃! u, ∀ x, (x ∈ u ↔ ∃ t ∈ s, x ∈ t) -- 和集合公理
theorem sigma₁_replacement {f : V → V} (hf : Σ₁-Function₁ f) (s : V) :
    ∃! t, ∀ y, (y ∈ t ↔ ∃ x ∈ s, y = f x) -- (制限された)置換公理図式
```

また有限の関数/有限列も扱える.

```
def IsMapping (m : V) : Prop := ∀ x ∈ domain m, ∃! y, ⟪x, y⟫ ∈ m
def Seq (s : V) : Prop := IsMapping s ∧ ∃ l, domain s = under l
```

3.6 原始再帰

$I\Sigma_1$では原始再帰法を用いて関数を定義できる.

定理 3.6.1: $f(\vec{v}), g(\vec{v}, x, z)$ を Σ_1 定義可能な関数とする. このとき, 以下を満たす Σ_1定義可能関数 $\mathrm{PRec}_{f,g}(\vec{v}, x)$ が存在する.

$$\mathrm{PRec}_{f,g}(\vec{v}, 0) = f(\vec{v})$$
$$\mathrm{PRec}_{f,g}(\vec{v}, x+1) = g\big(\vec{v}, x, \mathrm{PRec}_{f,g}(\vec{v}, x)\big)$$

```
structure Blueprint (k : ℕ) where
  zero : Σ₁-Semisentence (k + 1)
  succ : Σ₁-Semisentence (k + 3)
structure Construction {k : ℕ} (φ : Blueprint k) where
  zero : (Fin k → V) → V
  succ : (Fin k → V) → V → V → V
  zero_defined : DefinedFunction zero p.zero
  succ_defined : DefinedFunction (fun v ↦ succ (v ·.succ.succ) (v 1) (v 0)) p.succ
...
variable {k : ℕ} {φ : Blueprint k} (c : Construction V φ) (v : Fin k → V)
def Construction.result (u : V) : V
theorem Construction.result_zero :
    c.result v 0 = c.zero v
theorem Construction.result_succ (u : V) :
    c.result v (u + 1) = c.succ v u (c.result v u)
```

3.7 再帰的定義

定理 3.7.1: $\Phi_{\boldsymbol{C}}(\vec{v}, x)$ を**クラス $\boldsymbol{C}$** をパラメータとして取る述語だとする． Φ が以下を満たすならば,

1. 述語 $P(c, \vec{v}, x) := \Phi_{\{z \mid z \in c\}}(\vec{v}, x)$ は Δ_1 定義可能.
2. 単調： $\boldsymbol{C} \subseteq \boldsymbol{C}'$ ならば， $\Phi_{\boldsymbol{C}}(\vec{v}, x) \Longrightarrow \Phi_{\boldsymbol{C}'}(\vec{v}, x)$
3. 有限： $\Phi_{\boldsymbol{C}}(\vec{v}, x)$ ならば，ある m が存在して $\Phi_{\{z \in \boldsymbol{C} \mid z < m\}}(\vec{v}, x)$

次を満たす Σ_1 定義可能な述語 $\mathrm{Fix}_{\Phi}(\vec{v}, x)$ が存在する.

$$\mathrm{Fix}_{\Phi}(\vec{v}, x) \Longleftrightarrow \Phi_{\{x \mid \mathrm{Fix}_{\Phi}(\vec{v}, x)\}}(\vec{v}, x)$$

さらに次を満たすならば，$\mathrm{Fix}_{\Phi}(\vec{v}, x)$ は Δ_1 定義可能でその構造帰納法が証明できる.

4. 強有限： $\Phi_{\boldsymbol{C}}(\vec{v}, x) \Longrightarrow \Phi_{\{y \in \boldsymbol{C} \mid y < x\}}(\vec{v}, x)$

Fix_Φ を用いることで帰納的に定義されたクラス（リスト，形式化された項，形式化された論理式，形式化された証明, …）を定義できる．

```
structure Blueprint (k : ℕ) where
  core : Δ₁-Semisentence (k + 2)

structure Construction (φ : Blueprint k) where
  Φ : (Fin k → V) → Set V → V → Prop
  defined : Arith.Defined (fun v ↦ Φ (v ·.succ.succ) {x | x ∈ v 1} (v 0)) φ.core
  monotone {C C' : Set V} (h : C ⊆ C') {v x} : Φ v C x → Φ v C' x

class Construction.Finite (c : Construction V φ) where
  finite {C : Set V} {v x} : c.Φ v C x → ∃ m, c.Φ v {y ∈ C | y < m} x

class Construction.StrongFinite (c : Construction V φ) where
  strong_finite {C : Set V} {v x} : c.Φ v C x → c.Φ v {y ∈ C | y < x} x
...
variable {k : ℕ} {φ : Blueprint k} (c : Construction V φ) [Finite c] (v : Fin k → V)

def Construction.Fixpoint (x : V) : Prop

theorem Construction.case :
    c.Fixpoint v x ↔ c.Φ v {z | c.Fixpoint v z} x

theorem induction [c.StrongFinite] {P : V → Prop} (hP : Γ-[1]-Predicate P)
    (H : ∀ C : Set V, (∀ x ∈ C, c.Fixpoint v x ∧ P x) → ∀ x, c.Φ v C x → P x) :
    ∀ x, c.Fixpoint v x → P x
```

4. メタ数学の算術化

4.1 項のコーディング

Δ_1述語 T_C を以下のように定める.

$$u \in T_C \iff (\exists z)\left[u = \widehat{\#z}\right] \vee (\exists x)\left[u = \widehat{\&x}\right] \vee$$
$$(\exists k, f, v)\left[\mathrm{Func}(k, f) \wedge \mathrm{len}(v) = k \wedge (\forall i < k)[\mathrm{nth}(v, i) \in \boldsymbol{C}] \wedge u = \widehat{f^k(v)}\right]$$

$$\text{bounded variable:} \quad \widehat{\#z} := \langle 0,\, z\rangle + 1$$
$$\text{free variable:} \quad \widehat{\&z} := \langle 1,\, z\rangle + 1$$
$$\text{function:} \quad \widehat{f^k(v)} := \langle 2,\, f,\, k,\, v\rangle + 1$$

L を形式化された言語だとする． fixpoint construction により T_C の不動点 `L.IsUTerm : V → Prop` が得られる． また， その構造帰納法から束縛変数の最大値+1 を返す Σ_1関数 `L.termBV : V → V` が定義でき， `t ∈ V` がコード化された（n 個の束縛変数を持つ）擬項であることを表す Δ_1 述語 `L.IsSemiterm n t : Prop` が定義される.

```
def Language.IsSemiterm (n t : V) : Prop := L.IsUTerm t ∧ L.termBV t ≤ n
```

4.2 論理式のコーディング

Δ_1述語 $F_{\boldsymbol{C}}$ を以下のように定める.

$$
\begin{aligned}
u \in F_{\boldsymbol{C}} \iff & (\exists k, R, v)\Big[\mathrm{Rel}(k, R) \land u \text{ is a list of UTerm of length } k \land u = \widehat{R^k(v)}\Big] \lor \\
& (\exists k, R, v)\Big[\mathrm{Rel}(k, R) \land u \text{ is a list of UTerm of length } k \land u = \widehat{\neg R^k(v)}\Big] \lor \\
& u = \hat{\top} \lor u = \hat{\bot} \lor \\
& (\exists p, q \in \boldsymbol{C})\big[u = \widehat{p \land q}\big] \lor (\exists p, q \in \boldsymbol{C})\big[u = \widehat{p \lor q}\big] \lor \\
& (\exists p \in \boldsymbol{C})\big[u = \widehat{\forall p}\big] \lor (\exists p \in \boldsymbol{C})\big[u = \widehat{\exists p}\big]
\end{aligned}
$$

$$
\begin{aligned}
&\widehat{R^k(v)} := \langle 0,\, R,\, k,\, v\rangle + 1 \quad && \hat{\top} := \langle 2,\, 0\rangle + 1 \quad && \widehat{p \land q} := \langle 4,\, p,\, q\rangle + 1 \quad && \widehat{\forall p} := \langle 6,\, p\rangle + 1 \\
&\widehat{\neg R^k(v)} := \langle 1,\, R,\, k,\, v\rangle + 1 \quad && \hat{\bot} := \langle 3,\, 0\rangle + 1 \quad && \widehat{p \lor q} := \langle 5,\, p,\, q\rangle + 1 \quad && \widehat{\exists p} := \langle 7,\, p\rangle + 1
\end{aligned}
$$

擬項のときと同様にして形式化された擬論理式を指す Δ_1述語 `IsSemiformula n p : Prop` が定義される.

4.3 Tait 計算のコーディング

T を Δ_1 定義可能な理論とする． Δ_1述語 D_C^T を以下のように定める．

$$
\begin{aligned}
d \in D_C^T \iff & (\forall p \in \mathrm{sqt}(d))[\mathrm{Semiformula}(0, p)] \wedge \\
& \Big[(\exists s, p)\big[d = \mathrm{AXL}(s, p) \wedge p \in s \wedge \hat{\neg} p \in s\big] \\
& (\exists s)\Big[d = \top\text{-}\mathrm{INTRO}(s) \wedge \hat{\top} \in s\Big] \vee \\
& (\exists s, p, q)\big(\exists d_p, d_q \in \boldsymbol{C}\big)\Big[d = \wedge\text{-}\mathrm{INTRO}\big(s, p, q, d_p, d_q\big) \wedge \widehat{p \wedge q} \in s \wedge \mathrm{sqt}\big(d_p\big) = s \cup \{p\} \wedge \mathrm{sqt}\big(d_q\big) = s \cup \{q\}\Big] \vee \\
& (\exists s, p, q)(\exists d \in \boldsymbol{C})\big[d = \vee\text{-}\mathrm{INTRO}(s, p, q, d) \wedge \widehat{p \vee q} \in s \wedge \mathrm{sqt}(d) = s \cup \{p, q\}\big] \vee \\
& (\exists s, p)(\exists d \in \boldsymbol{C})\Big[d = \forall\text{-}\mathrm{INTRO}(s, p, d) \wedge \widehat{\forall p} \in s \wedge \mathrm{sqt}(d) = s^+ \cup \Big\{p^+\Big(\widehat{\&0}\Big)\Big\}\Big] \vee \\
& (\exists s, p, t)(\exists d \in \boldsymbol{C})\Big[d = \exists\text{-}\mathrm{INTRO}(s, p, t, d) \wedge \widehat{\exists p} \in s \wedge \mathrm{sqt}(d) = s \cup \{p(t)\}\Big] \vee \\
& (\exists s)(\exists d' \in \boldsymbol{C})[d = \mathrm{WK}(s, d') \wedge s \supseteq \mathrm{sqt}(d')] \vee \\
& (\exists s)(\exists d' \in \boldsymbol{C})\Big[d = \mathrm{SHIFT}(s, d') \wedge s = \mathrm{sqt}(d')^+\Big] \vee \\
& (\exists s, p)(\exists d_1, d_2 \in \boldsymbol{C})\big[d = \mathrm{CUT}(s, p, d_1, d_2) \wedge \mathrm{sqt}(d_1) = s \cup \{p\} \wedge \mathrm{sqt}(d_2) = s \cup \{\hat{\neg} p\}\big] \vee \\
& (\exists s, p)[d = \mathrm{ROOT}(s, p) \wedge p \in s \wedge p \in T]\Big]
\end{aligned}
$$

$$\mathrm{sqt}(d) := \pi_1(d-1)$$
$$\mathrm{AXL}(s,p) := \langle s,\, 0,\, p\rangle + 1$$

$$\top\text{-INTRO}(s) := \langle s,\, 1,\, 0\rangle + 1$$
$$\wedge\text{-INTRO}(s,p,q,d_p,d_q) := \langle s,\, 2,\, p,\, q,\, d_p,\, d_q\rangle + 1$$
$$\vee\text{-INTRO}(s,p,q,d) := \langle s,\, 3,\, p,\, q,\, d\rangle + 1$$
$$\forall\text{-INTRO}(s,p,d) := \langle s,\, 4,\, p,\, d\rangle + 1$$
$$\exists\text{-INTRO}(s,p,t,d) := \langle s,\, 5,\, p,\, t,\, d\rangle + 1$$

$$\mathrm{WK}(s,d) := \langle s,\, 6,\, d\rangle + 1$$
$$\mathrm{SHIFT}(s,d) := \langle s,\, 7,\, d\rangle + 1$$
$$\mathrm{CUT}(s,p,d_1,d_2) := \langle s,\, 8,\, p,\, d_1 d_2\rangle + 1$$
$$\mathrm{ROOT}(s,p) := \langle s,\, 9,\, p\rangle + 1$$

D_C^T の不動点を取って `T.Derivation` とする．以下のように定める．

```
def Language.Theory.Derivable (T) (s : V) : Prop := ∃ d, T.DerivationOf d s
def Language.Theory.Provable (T) (p : V) : Prop := T.Derivable {p}
```

定義より `T.Derivable` や `T.Provable` は Σ_1.

4.4 Gödel 数化

（メタ論理の）項/論理式から（V の内部の）形式化された項/論理式への翻訳 $t \mapsto \lceil t \rceil, \varphi \mapsto \lceil \varphi \rceil$ をコーディング通りに定義する.

```
#eval Encodable.encode ("⊤" : Sentence 𝓛ₒᵣ)
  /- 7 -/
#eval Encodable.encode ("1 + 1 = 2" : Sentence 𝓛ₒᵣ)
  /- 28112830254219990177127521847052876827659336521831253478899248392447025579092574801493036620226056829379396867558990018685021300411793862047857546098162625163543635122202320614830504503299423716674457604847097724679081530932429177719145500724823024785245221897296791878960465491237275698575524824256219345311259281018522948935497856808093108095231766345841458443025403080631492029306853223861326740190114231247862748245410665797364948572233034830050682843058606898062610930694694651093315927548105552403458330915905448819137494694143637528618224729344358569993199414554694157607604695541475472075817435246300226348972516312172781370490000720142817763639758008432269310312579231167553225000584223488849283104065326226084742908739192628263984831945022926699541741313385904048253030714974625421453613623461228387816858454401402027394522864925228948323195026505752820542708189648988786860300699947274381590667487806793436703922580916715270838972818987012192890409519287109955207836968510432355251110156455584812736340110558525810941713371474413319589488840965955367236035713731862100996111697414690100593579289825171746500819078956565063353817595634565097605705845515007215881579535384687716598790454332214986079146396756433767753945159179390970536434321513384938452673431025360061852901163563007256844744737871410514819114528 5 -/
```

5. Hilbert-Bernays-Löb の可証性条件

5.1 可証性条件

補題 5.1.1: T を $\mathsf{I}\Sigma_1$ より強い理論，U を R_0 より強い Δ_1 定義可能な理論だとする．このとき，

D1 $U \vdash \sigma \Longrightarrow T \vdash \mathrm{Provable}_U(\ulcorner\sigma\urcorner)$

D2 $T \vdash \mathrm{Provable}_U(\ulcorner\sigma \to \tau\urcorner) \to \mathrm{Provable}_U(\ulcorner\sigma\urcorner) \to \mathrm{Provable}_U(\ulcorner\tau\urcorner)$

D3 $T \vdash \mathrm{Provable}_U(\ulcorner\sigma\urcorner) \to \mathrm{Provable}_U(\ulcorner\mathrm{Provable}_U(\ulcorner\sigma\urcorner)\urcorner)$

$\mathbb{N} \vDash T$ ならば[1] 更に次が成り立つ．

D1′ $U \vdash \sigma \Longleftrightarrow T \vdash \mathrm{Provable}_U(\ulcorner\sigma\urcorner)$

[1] 実際には Σ_1 健全性で十分．

5.1 可証性条件

D1 及び D2, D1′ は形式化された証明の性質を地道に証明すれば示せる.

```
theorem provableₐ_D1
    [𝐈𝚺₁ ≼ T] [U.Delta1Definable] {σ : SyntacticFormula L} :
    U ⊢! σ ➝ T ⊢! U.bewₐ σ

theorem provableₐ_D2
    [𝐈𝚺₁ ≼ T] [U.Delta1Definable] {σ π : SyntacticFormula L} :
    T ⊢! U.bewₐ (σ ➝ π) ➝ U.bewₐ σ ➝ U.bewₐ π

theorem provableₐ_complete
    [𝐈𝚺₁ ≼ T] [ℕ ⊧ₘ* T] [𝐑₀ ≼ U] {σ : LO.FirstOrder.Sentence ℒₒᵣ} :
    U ⊢! σ ⭤ T ⊢! U.bewₐ σ
```

12

[1] T ⊢ φ は φ の理論 T-証明の型, T ⊢! φ はそのような証明が存在することを表す命題.

[2] T.Provable p は p ∈ V に関するモデル上の形式化された証明可能性を表す. 一方 T.bewₐ σ は T.Provable ⌜σ⌝ を定義する論理式を指す.

5.2 形式化された Σ_1-完全性

D3 は直接示すのは難しいが，次の補題より従う．

補題 5.2.1 (形式化された Σ_1-完全性): T を $\mathsf{I}\Sigma_1$ より強い理論，U を R_0 より強い Δ_1定義可能な理論だとする．文 σ が Σ_1論理式ならば，次が証明可能．

$$T \vdash \sigma \to \mathrm{Provable}_U(\ulcorner \sigma \urcorner)$$

証明 T のモデル V の内部で作業する．

次を示せばよい：すべての Σ_1論理式 $\varphi(x_1, ..., x_k), a_1, ..., a_k \in V$ について，$V \vDash \varphi[a_1, ..., a_k] \Longrightarrow \mathrm{Provable}_U(\ulcorner \varphi \urcorner(\overline{a_1}, ..., \overline{a_k}))$ これは φ に関する帰納法で示せる．□

```
theorem provableₐ_sigma₁_complete
    {σ : Sentence ℒₒᵣ} (hσ : Hierarchy Σ 1 σ) :
    T ⊢! σ → U.bewₐ σ
```

6. 不完全性定理

6.1 第一不完全性定理

G1 は 1, 2, 3, 4 までの結果で証明できる.

定理 6.1.1 (G1): T が Δ_1 定義可能で R_0 より強く Σ_1-健全ならば, T から証明も反証もできない論理式が存在する.

証明 $D := \{\ulcorner\varphi\urcorner \mid \varphi : 1\text{ 変数の論理式}, T \vdash \neg\varphi(\ulcorner\varphi\urcorner)\}$ と定義する. $T \vdash \pi \Leftrightarrow \mathbb{N} \vDash \mathrm{Provable}_T(\ulcorner\pi\urcorner)$, また $\mathrm{Provable}_T$ は Σ_1定義可能なので, D は r.e.集合.

表現定理より次を満たす θ が存在する: すべての $n \in \mathbb{N}$ について $n \in D \iff T \vdash \theta(\overline{n})$.

$n = \ulcorner\theta\urcorner$ と置くと,

$$T \vdash \theta(\ulcorner\theta\urcorner) \iff \ulcorner\theta\urcorner \in D \iff T \vdash \neg\theta(\ulcorner\theta\urcorner)$$

よって T が完全だと仮定すると矛盾する. □

6.1 第一不完全性定理

```
theorem goedel_first_incompleteness : ¬System.Complete T := by
  let D : ℕ → Prop := fun n : ℕ ↦ ∃ p : SyntacticSemiformula ℒₒᵣ 1, n = ⌜p⌝ ∧ T ⊢! ~p/[⌜p⌝]
  -- D の定義
  have D_re : RePred D := by ...
  -- D は r.e.
  let θ : SyntacticSemiformula ℒₒᵣ 1 := codeOfRePred (D)
  have : ∀ n : ℕ, D n ↔ T ⊢! θ/[‘↑n’] := fun n ↦ by
    simpa [Semiformula.coe_substs_eq_substs_coe₁] using re_complete (T := T) (D_re) (x := n)
  -- D を表現する論理式 θ を取る
  let ρ : SyntacticFormula ℒₒᵣ := θ/[⌜θ⌝]
  -- ρ := θ(⌜θ⌝)
  have : T ⊢! ~ρ ↔ T ⊢! ρ := by
    simpa [D, goedelNumber'_def, quote_eq_encode] using this ⌜θ⌝
  -- T ⊢ ¬ρ ↔ T ⊢ ρ
  have con : System.Consistent T := consistent_of_sigma1Sound T
  -- T は無矛盾
  refine LO.System.incomplete_iff_exists_undecidable.mpr ⟨↑ρ, ?_, ?_⟩
  -- ρ が T から独立であることを示す
  · intro h
    have : T ⊢! ~↑ρ := by simpa [provable₀_iff] using this.mpr h
    exact LO.System.not_consistent_iff_inconsistent.mpr (inconsistent_of_provable_of_unprovable h this) inferInstance
    -- T ⊢ ¬ρ ならば矛盾.
  · intro h
    have : T ⊢! ↑ρ := this.mp (by simpa [provable₀_iff] using h)
    exact LO.System.not_consistent_iff_inconsistent.mpr (inconsistent_of_provable_of_unprovable this h) inferInstance
    -- T ⊢ ρ ならば矛盾.
```

6.2 第二不完全性定理

補題 6.2.1 (不動点補題): 1 変数の論理式 θ について，次を満たす 文 $\mathrm{fixpoint}_\theta$ が存在する.

$$\mathsf{I}\Sigma_1 \vdash \mathrm{fixpoint}_\theta \leftrightarrow \theta(\ulcorner \mathrm{fixpoint}_\theta \urcorner)$$

証明 形式化された論理式に数項を代入する関数のグラフ $\mathrm{substNumeral}(y, p, x)$ は Σ_1 定義可能.

$$\mathsf{I}\Sigma_1 \vdash \mathrm{substNumeral}(y, p, x) \leftrightarrow y = p(\overline{x})$$

$\mathrm{fixpoint}_\theta$ を次のように定義する.

$$\begin{aligned} \mathrm{diag}_\theta(x) &:= (\forall y)[\mathrm{substNumeral}(y, x, x) \to \theta(y)] \\ \mathrm{fixpoint}_\theta &:= \mathrm{diag}_\theta(\ulcorner \mathrm{diag}_\theta \urcorner) \end{aligned}$$

このとき，$\mathsf{I}\Sigma_1$ に於いて,

$$\begin{aligned} \mathrm{fixpoint}_\theta &\overset{\text{def}}{=} (\forall y)[\mathrm{substNumeral}(y, \ulcorner \mathrm{diag}_\theta \urcorner, \ulcorner \mathrm{diag}_\theta \urcorner) \to \theta(y)] \\ &\leftrightarrow \theta(\ulcorner \mathrm{diag}_\theta(\ulcorner \mathrm{diag}_\theta \urcorner) \urcorner) \\ &\overset{\text{def}}{=} \theta(\ulcorner \mathrm{fixpoint}_\theta \urcorner) \end{aligned}$$

□

6.2 第二不完全性定理

Gödel 文「この文は証明できない」を定義する． $\mathrm{G}_T := \mathrm{fixpoint}_{\neg \mathrm{Provable}_T}$

以降 T を $\mathsf{I}\Sigma_1$ より強い理論とする．

補題 6.2.2:
1. T が無矛盾ならば $T \nvdash \mathrm{G}_T$,
2. $\mathbb{N} \vDash T$ ならば $T \nvdash \neg \mathrm{G}_T$.

証明

1. $T \vdash \mathrm{G}_T$ と仮定する．不動点補題より $T \vdash \neg \mathrm{Provable}_T(\ulcorner \mathrm{G}_T \urcorner)$を得る． 一方 D1 より $T \vdash \mathrm{Provable}_T(\ulcorner \mathrm{G}_T \urcorner)$ なので T は矛盾する． これは仮定に反する．
2. $T \vdash \neg \mathrm{G}_T$ と仮定する． 不動点補題より $T \vdash \mathrm{Provable}_T(\ulcorner \mathrm{G}_T \urcorner)$ を得る． D1′ より $T \vdash \mathrm{G}_T$ なので T は矛盾する． これは仮定に反する．

6.2 第二不完全性定理

T の無矛盾性を表す文を定義する． $\mathrm{Con}_T := \neg\mathrm{Provable}_T(\ulcorner\bot\urcorner)$

補題 6.2.3: $T \vdash \mathrm{Con}_T \leftrightarrow \mathrm{G}_T$

証明

(→) $\neg\mathrm{G}_T \to \mathrm{Provable}_T(\ulcorner\bot\urcorner)$ を示せばよい． $\neg\mathrm{G}_T$ を仮定する．不動点補題より$\mathrm{Provable}_T(\ulcorner\mathrm{G}_T\urcorner)$. また，不動点補題と D1 より $\mathrm{Provable}_T(\ulcorner\mathrm{G}_T \to \neg\mathrm{Provable}_T(\ulcorner\mathrm{G}_T\urcorner)\urcorner)$. よって D2 より $\mathrm{Provable}_T(\ulcorner\neg\mathrm{Provable}_T(\ulcorner\mathrm{G}_T\urcorner)\urcorner)$.

一方 D3 より $\mathrm{Provable}_T(\ulcorner\mathrm{Provable}_T(\ulcorner\mathrm{G}_T\urcorner)\urcorner)$. 再び D2 を用いて $\mathrm{Provable}_T(\ulcorner\bot\urcorner)$ を得る．

(←) $\mathrm{Provable}_T(\ulcorner\bot\urcorner) \to \neg\mathrm{G}_T$ を示せばよい． $\mathrm{Provable}_T(\ulcorner\bot\urcorner)$ を仮定する．D1 より $\mathrm{Provable}_T(\ulcorner\bot\to \mathrm{G}_T\urcorner)$, よって D2 より $\mathrm{Provable}_T(\ulcorner\mathrm{G}_T\urcorner)$. 不動点補題より $\neg\mathrm{G}_T$ を得る． □

定理 6.2.1 (G2): T が無矛盾ならば $T \nvdash \mathrm{Con}_T$. $\mathbb{N} \vDash T$ ならば $T \nvdash \neg\mathrm{Con}_T$.

6.2 第二不完全性定理

よって以下が証明できる.

```
theorem goedel_second_incompleteness
    [𝐈𝚺₁ ≼ T] [T.Delta1Definable] [LO.System.Consistent T] :
    T ⊬ ↑Con

theorem inconsistent_undecidable
    [𝐈𝚺₁ ≼ T] [T.Delta1Definable] [ℕ ⊨ₘ* T] :
    System.Undecidable T ↑Con
```

ただし，上の証明が本質的に依存しているのは D1, D2, D3 と述語の健全性のみ.

6.3 証明可能性論理

以下の内容は私（齋藤）ではなく，主に神戸大の野口氏の形式化した結果である．

型クラスを用いて証明可能性述語や Hilbert-Bernays の可証性条件を抽象的に定義できる．

```
structure ProvabilityPredicate (T : Theory L) (U : Theory L) where
  prov : Semisentence L 1
  spec {σ : Sentence L} : U ⊢! σ → T ⊢! prov/[⌜σ⌝] -- 少なくとも D1 を満たす

class Diagonalization (T : Theory L) where
  fixpoint : Semisentence L 1 → Sentence L
  diag (θ) : T ⊢!. fixpoint θ ↔ θ/[⌜fixpoint θ⌝]

...

class ProvabilityPredicate.HBL2 (𝔅 : ProvabilityPredicate T U) where
  D2 {σ τ : Sentence L} : T ⊢! 𝔅 (σ → τ) → (𝔅 σ) → (𝔅 τ) -- D2

class ProvabilityPredicate.HBL3 (𝔅 : ProvabilityPredicate T U) where
  D3 {σ : Sentence L} : T ⊢! (𝔅 σ) → 𝔅 (𝔅 σ) -- D3

class ProvabilityPredicate.HBL (𝔅 : ProvabilityPredicate T₀ T)
  extends 𝔅.HBL2, 𝔅.HBL3 -- D1 + D2 + D3
```

6.3 証明可能性論理

これらの仮定の上で一般的に G1 や G2 が証明できる.

第一不完全性定理：

```
theorem goedel_independent
    [T ≤ U] [Diagonalization T] [LO.System.Consistent U]
    (𝔅 : ProvabilityPredicate T U) [𝔅.GoedelSound] :
    System.Undecidable U (goedel 𝔅)
```

第二不完全性定理：

```
theorem unprovable_consistency
    [T ≤ U] [Diagonalization T] [System.Consistent U]
    (𝔅 : ProvabilityPredicate T U) [𝔅.HBL] :
    U ⊬ con 𝔅
```

6.4 今後

- 算術的完全性定理.
- Paris-Harrington の定理等の独立命題に関する結果.
- 集合論.
- 二階算術.
- 直観主義一階述語論理, 特に Heyting arithmetic.
- Büchi arithmetic や S2S の決定性.
- 自動証明.

⋮

Bibliography

[1] L. C. Paulson, “A mechanised proof of Gödel’s incompleteness theorems using Nominal Isabelle,” Journal of Automated Reasoning, vol. 55, pp. 1–37, 2015.

[2] W. Pohlers, Proof theory: The first step into impredicativity. Springer Science & Business Media, 2008.

[3] P. Hájek and P. Pudlák, Metamathematics of first-order arithmetic, vol. 3. Cambridge University Press, 2017.

[4] S. Cook and P. Nguyen, Logical foundations of proof complexity, vol. 11. Cambridge University Press Cambridge, 2010.

Sponsor

FormalizedFormalLogic is supported by Proxima Technology Proxima