

Mechanizing Gödel's Incompleteness Theorems and Provability Logic

定理証明支援系 Lean による 不完全性定理・証明可能性論理の形式化について

野口 真柊 | 神戸大学システム情報学研究科 M2

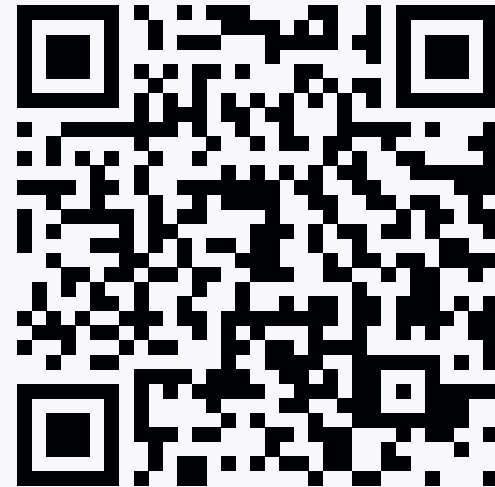
2026/10/07 @ SLACS 2026



1. はじめに

この発表では定理証明支援系 Lean で Gödel の不完全性定理と証明可能性論理に関する事実を形式化したということを発表します。厳密に話すと時間がいくらあっても足りないので、余談を挟みつつラフに解説していきます（スライドも飛ばし飛ばしで使います）。

- 最新版のスライド: <https://sno2wman.github.io/slides-for-MGITaPL/main.pdf>
- プレプリント: <https://arxiv.org/abs/2609.13780>
- 実装: <https://github.com/FormalizedFormalLogic/Foundation>



注意：プレプリント出版時とリポジトリの構成が大きく異なっています（Foundation へ統合しています）

2. 定理証明支援系

要約

数学の実践において、人はしばしば論理の運行を間違える． 例: 仮定の不足, 帰納法, etc.

→ **機械に検証させればいいじゃん！**

定理証明支援系とは数学をプログラミングコード（形式言語）として記述し、その記述が論理的に破綻していないことを検証するソフトウェアである．

例: Rocq, Isabelle, Agda, HOL Light, **Lean**

2020 年に新しく開発された定理証明支援系.

Lean 4 は型理論として Calculus of Inductive (CIC) を採用しており, 原理上は¹普通に行われている数学が全部展開出来るだろうとされている.

特徴

- ・強力なメタプログラミング能力 (マクロや新しいタクティクの作成が容易).
- ・汎用的な数学的ライブラリ Mathlib (代数, 解析学, 圏論, etc.²).
- ・実用的なプログラミング言語への志向.

近年 AI/LLM が寄越してきた証明の保証として用いられる風潮がある³.

¹Lean 3 では Carneiro (2019) が $ZFC + \lceil \omega\text{-個の到達不能基数が存在する} \rceil$ の無矛盾性をモデルを作って示している. この結果を素直にこの結果を Lean 4 に持ってくることは出来ない (らしい) が, 最近ようやく <https://github.com/leanprover/con-leche> などに取り組まれているように思える. もちろん今ある数学が全然 ZFC でやってるわけねーだろという方にとっては一切この話は関係ない.

²学部レベルの数学は全部あるらしい.

³なぜなのかはよくわからない. コンパイルの速度?

Gödel の不完全性定理の形式化の先行研究として例えば以下がある.

- Shankar (1986) による Boyer–Moore 定理証明器 (Nqthm)
 - G1 の最初の形式化.
- O'Connor (2005) による Rocq での G1, G2 の形式化.
 - G1 は PA ではなくて Z2 という算術体系上で, G2 はいくつかの仮定を公理として.
- Paulson (2014) による G2 の形式化
 - 唯一の G2 の形式化だが, 遺伝的有限集合論で議論.

1 つ目の新規性: フルの算術で全部やった.

様相論理 GL の形式化として

- Maggesi and Perini Brogi (2023) でラベル付きシーケント計算の HOL Light での形式化.

2つ目の新規性：実際に GL の様相を算術に関連付けた形式化は誰もやっていない.

余談: **論理学の形式化はそれほど活発に行われていない！**¹

¹なぜかはよくわからない. 単に数学基礎論がそんなに流行ってないという分野の問題かもしれないし, 方向性としてこの分野の研究者が伝統的な記号論理よりも型理論の方に行ってしまうからなのかもしれない.

最近のニュース: Ramana Kumar¹ は Collatz 予想の反証を Lean v4.32.1 で形式化した.

```
-- `n` はCollatzの操作を何度行っても1にならない.  
def Diverges (n : Nat) : Prop := 0 < n ∧ ∀ k, iterate step k n ≠ 1  
  
-- そのような `n` が存在する.  
theorem exists_nonterminating_orbit : ∃ n, Diverges n :
```

もちろん²このような上手い話があるわけがなく, これは Lean のソフトウェアとしての実装のバグに由来するものであった (詳しい解説は Moura (2026), v4.33.0 で修正).

甚大な量の Lean による形式証明のソースコードの中にこのような秘孔が混じっている可能性はある. これに関しては Section 5.5 でも論ずる.

¹<https://github.com/xrchz/CollatzLean>

²この講演が行われた当時は少なくとも

ひとくちに定理証明支援系を使うと言っても、次のことに目を向けるべきだと感じる¹.

注意 1 (定理証明支援系による正しさの公準): 定理証明支援系が/の/で...

1. 依拠している数学的基礎に問題や矛盾が無いのか？ (論理学者として)
2. ソフトウェアとしての実装にバグが存在しないか？ (ソフトウェア開発者として)
3. 形式化したそのコードは命題を正しく形式化しているか？ (エンドユーザとして)

¹この基準は @Alwe_Logic (2026) のツイートから拝借した.

3.1 階述語論理および不完全性定理

注意 2: formalize という語の意図を以下で使い分ける.

formalize (形式化) 数理論理学の技法として数学的議論を形式操作だと思って展開する.

mechanize (機械化) 定理証明支援系によって数学をプログラムによって実装する.

我々がやったことの端的な説明：**数学の形式化の機械化！**

次の Gödel の不完全性定理の素朴なバージョンを機械化した.

定理 (Gödel の第 1 不完全性定理(G1)): T が Cobham の最弱の算術 R_0 を含み, Δ_1 -定義可能¹ で, Σ_1 -健全なら, T から証明も反証も出来ない文が存在する.

定理 (Gödel の第 2 不完全性定理(G2)): T が $|\Sigma_1$ より強く, Δ_1 -定義可能で, 無矛盾なら, T の無矛盾性を表す文は証明できない.

いくつかの条件は改良できる(後述).

```
theorem incomplete (T : ArithmeticTheory) [T. $\Delta_1$ ] [ $R_0 \leq T$ ] [T.SoundOnHierarchy  $\Sigma$  1] : Incomplete T

theorem consistent_unprovable [Consistent T] : T  $\not\vdash$  T.consistent.val
```

Source: [1](#) [2](#)

¹ T の公理を記述する論理式が Δ_1 -論理式で記述出来る

論理式 (擬論理式) は否定標準形で扱う. L は言語, 型 ξ を自由変数, 束縛変数は de Bruijn インデックスによって自然数 $\mathbb{N}$ で扱うこととする.

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

```
inductive Semiformula (L : Language) (ξ : Type*) : ℕ → Type _ where
| verum : Semiformula L ξ 0
| falsum : Semiformula L ξ 0
| rel : {arity : ℕ} → L.Rel arity → (Fin arity → Semiterm L ξ n) → Semiformula L ξ n
| nrel : {arity : ℕ} → L.Rel arity → (Fin arity → Semiterm L ξ n) → Semiformula L ξ n
| and : Semiformula L ξ n → Semiformula L ξ n → Semiformula L ξ n
| or : Semiformula L ξ n → Semiformula L ξ n → Semiformula L ξ n
| all : Semiformula L ξ (n + 1) → Semiformula L ξ n
| exs : Semiformula L ξ (n + 1) → Semiformula L ξ n
```

Source: [1](#)

- Formula $L \ \xi$ を Semiformula $L \ \xi \ 0$ の略記 (束縛変数無し)
- Semisentence $L \ n$ を Semiformula $L \ \text{Empty} \ n$ の略記 (自由変数無し)
- Sentence L を Formula $L \ \text{Empty}$ の略記 (自由・束縛変数無し)

算術の言語 $\mathcal{L}_{\text{OR}}$ を定めて, Lean のマクロによる糖衣構文を用意する. 例えばこんな感じで記述出来る.

論理式	Lean
$98 + 6748 = 6846$	<code>"98 + 6748 = 6846"</code>
$\forall x. \forall y. a \cdot (x + y) = a \cdot x + a \cdot y$	<code>"a. ∀ x y, a * (x + y) = a * x + a * y"</code>
$\forall x. [x < \bar{n} \leftrightarrow \bigvee_{i < n} x = \bar{i}]$	<code>"∀ x, x < ↑n ↔ ∀ i < n, x = ↑i"</code>

ここで $a.$ は束縛変数 (ArithmeticSemisentence 1 の変数), $\uparrow n$ はメタの自然数 n の数項である.

古典 1 階述語論理の Tait 流のシーケント計算体系 LK を用意する. シーケントは多重集合として定義する.

$$\begin{array}{c}
 \frac{}{R(\vec{v}), \neg R(\vec{v}), \Delta} \text{id} \quad \frac{}{\top, \Delta} \text{verum} \quad \frac{\Gamma}{\Delta} \text{wk}^1 \quad \frac{\Gamma, \varphi, \varphi}{\Gamma, \varphi} \text{ctr} \quad \frac{\varphi, \Delta \quad \neg \varphi, \Delta}{\Delta} \text{cut} \\
 \frac{\varphi, \psi, \Delta}{\varphi \vee \psi, \Delta} \text{or} \quad \frac{\varphi, \Delta \quad \psi, \Delta}{\varphi \wedge \psi, \Delta} \text{and} \quad \frac{\varphi^+(\&0), \Delta^+}{\forall \varphi, \Delta} \text{all}^2 \quad \frac{\Gamma, \varphi \quad \varphi(t), \Delta}{\exists \varphi, \Delta} \text{ex}
 \end{array}$$

```

inductive LK.Derivation : LK.Sequent L → Type _
| identity (r : L.Rel k) (v) : LK.Derivation {ℓ.rel r v, .nrel r v}
| verum : LK.Derivation {ℓ.⊤}
| weakening : LK.Derivation Γ → LK.Derivation (Γ + {ℓ.φ})
| contraction : LK.Derivation (Γ + {ℓ.φ, φ}) → LK.Derivation (Γ + {ℓ.φ})
| cut : LK.Derivation (Γ + {ℓ.φ}) → LK.Derivation (Δ + {ℓ.¬φ}) → LK.Derivation (Γ + Δ)
| or : LK.Derivation (Γ + {ℓ.φ, ψ}) → LK.Derivation (Γ + {ℓ.φ ∨ ψ})
| and : LK.Derivation (Γ + {ℓ.φ}) → LK.Derivation (Γ + {ℓ.ψ}) → LK.Derivation (Γ + {ℓ.φ ∧ ψ})
| all : LK.Derivation (Γ+ + {ℓ.φ.free}) → LK.Derivation (Γ + {ℓ.∀1 φ})
| exs : LK.Derivation (Γ + {ℓ.φ/[t]}) → LK.Derivation (Γ + {ℓ.∃1 φ})

```

Source: [1](#)¹Γ ⊆ Δ²ここでは自由変数を &0, &1, ... 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

カット無しの証明図へ変換する具体的な計算手続きを定める¹ことで、LK ではカット除去定理を機械化出来る².

```
def hauptsatz {Γ : LK.Sequent L} : ⊢LK1 Γ → {d : ⊢LK1 Γ // LK.Derivation.IsCutFree d}
```

Source: [1](#)

言語 L の理論 T を L -文の集合 Set (Sentence L) とする.

- φ が T から導出できることを $T \vdash \varphi$ と書く.
- T を満たす任意のモデル M で φ も満たされるとき, $T \models \varphi$ と書く.

カット除去定理からカノニカルモデルを作るなどの議論を行って, 完全性定理を得る.

```
theorem small_satisfiable_of_consistent : Consistent T → Satisfiable T
```

```
theorem Proof.complete_iff : T ⊨ φ ↔ T ⊢ φ := ⟨fun h ↦ Proof.complete h, Proof.sound⟩
```

Source: [1](#)

¹証明図の帰納法による愚直な証明は機械化において煩雑で面倒なので, Avigad (2004), Avigad (2001) による直観主義述語論理への還元および強制法的な議論による.

²ただしこれが現実的に Lean で計算可能なかはわからない.

以降, 言語 $\mathcal{L}_{\text{OR}}$ ($\mathcal{L}_{\text{or}}$) の理論を算術と呼ぶ.

定義 3: Cobham の最弱の算術 R_0 は $\mathcal{L}_{\text{OR}}$ の等号公理と, 次の公理図式からなる理論である ($n, m \in \mathbb{N}$). $\bar{n}$ は数項を表す.

$$\bar{n} + \bar{m} = \overline{n + m}, \quad \bar{n} \cdot \bar{m} = \overline{n \cdot m}, \quad \forall x. \left[x < \bar{n} \leftrightarrow \bigvee_{i < n} x = \bar{i} \right]$$

```
inductive R0 : ArithmeticTheory
| equal      : ∀ φ ∈ EQ L_or, R0 φ
| Ω1 (n m : ℕ) : R0 "↑n + ↑m = ↑(n + m)"
| Ω2 (n m : ℕ) : R0 "↑n * ↑m = ↑(n * m)"
| Ω3 (n : ℕ)   : R0 "∀ x, x < ↑n ↔ ∀ i < n, x = ↑i"
```

```
notation "R0" ⇒ R0
```

定義 4: PA^- は離散順序半環の基本的な性質を述べる有限個の Π_1 文からなる理論である.

```

abbrev addZero      := "∀ x, x + 0 = x"
abbrev addAssoc     := "∀ x y z, (x + y) + z = x + (y + z)"
abbrev addComm      := "∀ x y, x + y = y + x"
abbrev addEqOfLt    := "∀ x y, x < y → ∃ z <+ y, x + z = y"
abbrev zeroLe       := "∀ x, 0 ≤ x"
abbrev zeroLtOne    := "0 < 1"
abbrev oneLeOfZeroLt := "∀ x, 0 < x → 1 ≤ x"
abbrev addLtAdd      := "∀ x y z, x < y → x + z < y + z"
abbrev mulZero       := "∀ x, x * 0 = 0"
abbrev mulOne        := "∀ x, x * 1 = x"
abbrev mulAssoc      := "∀ x y z, (x * y) * z = x * (y * z)"
abbrev mulComm       := "∀ x y, x * y = y * x"
abbrev mulLtMul      := "∀ x y z, x < y ∧ 0 < z → x * z < y * z"
abbrev distr         := "∀ x y z, x * (y + z) = x * y + x * z"
abbrev ltIrrefl      := "∀ x, x ≠ x"
abbrev ltTrans       := "∀ x y z, x < y ∧ y < z → x < z"
abbrev ltTri         := "∀ x y, x < y ∨ x = y ∨ x > y"

```

定義 5: 1 変数論理式 $\varphi(x)$ の帰納法 $\varphi(0) \rightarrow \forall x. [\varphi(x) \rightarrow \varphi(x+1)] \rightarrow \forall x. \varphi(x)$ の全称閉包を $\text{Ind}(\varphi)$ と書く.

- $\text{I}\Sigma_1 := \text{PA}^- + \{\text{Ind}(\varphi) : \varphi \text{ は冠頭標準形の } \Sigma_1\text{-論理式}\}.$
- $\text{PA} := \text{PA}^- + \{\text{Ind}(\varphi) : \varphi \text{ は任意の論理式}\}.$

```
def succInd {ξ} (φ : Semiformula L ξ 1) : Formula L ξ :=
  " !φ 0 → (∀ x, !φ x → !φ (x + 1)) → ∀ x, !φ x "
def InductionScheme (Γ : Semiformula L ℕ 1 → Prop) : Theory L :=
  { ψ | ∃ φ : Semiformula L ℕ 1, Γ φ ∧ ψ = .univCl (succInd φ) }

abbrev InductionOnPrenexHierarchy (Γ : Polarity) (s : ℕ) : ArithmeticTheory :=
  PA⁻ ∪ InductionScheme L_or (B[<, L_or].PrenexHierarchy Γ s)
prefix:max "IND " ⇒ InductionOnPrenexHierarchy

abbrev ISigma (s : ℕ) : ArithmeticTheory := IND Σ s
notation "IΣ₁" ⇒ ISigma 1

abbrev Peano : ArithmeticTheory := PA⁻ ∪ InductionScheme L_or Set.univ
notation "PA" ⇒ Peano
```

ここについての扱いは現在改修中.

現れる量化子が限定量化であるような論理式は Δ_0 -論理式と呼ぶ.

実用的な問題のため, 論理式が Σ_1 や Π_1 であるとは限定量化を途中に挟んでもよく, また Boolean 結合で閉じているものとする. 例えば以下も Σ_1 -論理式である.

$$\exists x, \forall y < t, \exists z. \varphi(x, y, z) \wedge \forall x < t. \psi(x)$$

一方で, 帰納法のクラスなどを Σ_n や Π_n で制限するときは冠頭標準形に制限する.

$$\exists \forall \exists \forall \dots \varphi(x_1, x_2, \dots, x_n, \vec{y})$$

冠頭標準形定理および採集原理などがあればこれらの区別は厳密にする必要はないが, 機械化ではこのような細かい議論や定義も必要になる.

$T \vdash \varphi$ であることを（特に Lean で機械化するには）実際に証明図を構成しなければならないが、それは人力では現実的ではない。

完全性定理を機械化することで以下の還元が使える。

$$\text{Lean} \vdash "T \vdash \varphi" \iff \text{Lean} \vdash "\forall V, V \models T \implies V \models \varphi"$$

意味論的な議論においては、例えば T が十分に豊かな算術であるなら $V \models T$ を満たす V が良い代数的な構造になる。Mathlib などが提供する代数的な構造に対しての様々な補題やメタプログラミング、自動証明タクティクが利用出来る。これを戻して $T \vdash \varphi$ を簡単に示せる。

算術 T の任意のモデル V を固定する.

```
variable {V : Type*} [ORingStructure V] [V↓[ $\mathcal{L}_{\text{OR}}$ ]  $\models^* T$ ]
```

- `ORingStructure V`: V が言語 $\mathcal{L}_{\text{OR}}$ の構造であることを主張する typeclass.
- `V↓[ $\mathcal{L}_{\text{OR}}$ ]  $\models^* T$` : V が理論 T を満たすことを主張する typeclass.

V 上で機械化を行う. 関数は選択関数を用いて定義出来る.

```
lemma sqrt_exists_unique (a : V) :  $\exists!$  x,  $x * x \leq a \wedge a < (x + 1) * (x + 1)$ 
```

```
noncomputable def sqrt (a : V) : V := Classical.choose! (sqrt_exists_unique a)  
prefix:75 "√"  $\Rightarrow$  sqrt
```

```
lemma sqrt_mul_self (a : V) :  $\sqrt{a * a} = a$ 
```

形式体系を算術の中でさらに形式化する¹: 算術化・Bootstrapping.

- 論理式 φ や導出木 D に対して V への割り当て「 $\cdot$ 」を割り当てる (Gödel 数).
- 逆に V の要素 x が項, 論理式, 導出木の Gödel 数であるというメタの (Lean 上の) 述語 $\text{IsFormula}(x) : V \mapsto 2$ など考える.

メタの述語 $\text{IsFormula}(x)$ に対応する $\mathcal{L}_{\text{OR}}$ -論理式 $\text{IsFormula}(\ulcorner \varphi \urcorner)$ などを $\Delta_0, \Sigma_n, \Pi_n$ の適当な階層でひたすら頑張って構成していく.

$$V \models \text{IsFormula}(\ulcorner \varphi \urcorner) \iff T \vdash \text{IsFormula}(\ulcorner \varphi \urcorner)$$

¹Formalizing [Formalizing [Formalizing mathematics in formal system] in Arithmetic] in Lean

R_0 では表現定理が成り立つ¹.

定理 6 (R_0 の表現定理): Σ_1 -健全な $T \supseteq R_0$ と, S を r.e.集合とする. このとき, $\mathcal{L}_{\text{OR}}$ -論理式 $\text{Rep}_S(x)$ があって以下を満たす.

$$n \in S \iff T \vdash \text{Rep}_S(\bar{n})$$

```
noncomputable def codeOfREPred (p : ℕ → Prop) : ArithmeticSemisentence 1

theorem rePred_weak_representation {p : ℕ → Prop} (hp : REPred p) {x : ℕ} :
  p x ↔ T ⊢ (codeOfREPred p)/[x]
```

Source: [1](#)

¹もちろん Robinson 算術 や PA でも成り立つが, R_0 は表現定理が成り立つよく知られている算術理論の中では最弱とされる.

表現定理および算術化を用いて, Gödel の第1不完全性定理(G1)をまず機械化出来る.

定理 7 (G1): T が Cobham の最弱の算術 R_0 を含み, Δ_1 -定義可能で, Σ_1 -健全なら, T から証明も反証も出来ない文が存在する.

証明: $D := \{ \ulcorner \varphi \urcorner : \varphi \text{ は 1 変数論理式かつ } T \vdash \neg \varphi(\ulcorner \varphi \urcorner) \}$ を取ると, これは r.e.なので $\theta(x)$ が存在して $n \in D \iff T \vdash \theta(\bar{n})$. 以下の同値性が成立する.

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

ゆえに T が完全なら無矛盾性に反する. ■

```
theorem incomplete (T : ArithmeticTheory) [T.Δ1] [R0 ≤ T] [T.SoundOnHierarchy Σ 1] : Incomplete T
```

Source: [1](#)

ここで Incomplete T は $\exists \varphi, T \not\vdash \varphi \wedge T \not\vdash \sim \varphi$ の略記.

生の証明可能性述語を機械化で直接扱うと面倒なので、抽象化を導入する。

定義 8 (証明可能性): 1 変数述語 $\mathfrak{B}$ が T_0 上の T -証明可能性であるとは, $D1 : T \vdash \sigma \implies T_0 \vdash \mathfrak{B} \sigma$ を任意の文 σ で満たすこととする. 追加条件として以下を満たすとき, **HBL** である¹ という.

- **D2** : $T_0 \vdash \mathfrak{B}(\sigma \rightarrow \tau) \rightarrow (\mathfrak{B} \sigma \rightarrow \mathfrak{B} \tau)$
- **D3** : $T_0 \vdash \mathfrak{B} \sigma \rightarrow \mathfrak{B}(\mathfrak{B} \sigma)$

```
structure Provability (T₀ : Theory L₀) (T : Theory L) where
  prov : Semisentence L₀ 1
  bew_def {σ : Sentence L} : T ⊢ σ → T₀ ⊢ prov/[⌈σ⌋]
```

```
class HBL where
  D2 {σ τ : Sentence L} : T₀ ⊢ B (σ → τ) → B σ → B τ
  D3 {σ : Sentence L} : T₀ ⊢ B σ → B (B σ)
```

Source: [1](#)

¹Hilbert-Bernays-Löb.

定義 9 (対角化可能性): T が対角化可能とは, 1 変数述語 θ を入力とし文 fixpoint_θ を返す関数があって, それは以下を満たす.

$$T \vdash \text{fixpoint}_\theta \leftrightarrow \theta(\ulcorner \text{fixpoint}_\theta \urcorner)$$

```
class Diagonalization [L.ReferenceableBy L] (T : Theory L) where
  fixedpoint : Semisentence L → Sentence L
  diag (θ) : T ⊢ fixedpoint θ ↔ θ/[⌊fixedpoint θ⌋]

theorem diagonal (θ : ArithmeticSemisentence → Sentence L) :
  T ⊢ fixedpoint θ ↔ θ/[⌊fixedpoint θ⌋]
```

定義 10:

- 無矛盾性を表す文 $\neg \mathcal{B} \perp$: 「矛盾は証明できない」を $\text{Con}_{\mathcal{B}}$ とする.
- $\neg \mathcal{B}(\cdot)$ の不動点を Gödel 文 $G_{\mathcal{B}}$ とする.

```
def con (B : Provability T₀ T) : Sentence L₀ := ~B ⊥
```

```
def gödel (B : Provability T₀ T) : Sentence L := fixedpoint T₀ “x. ~!B.prov x”
```

```
lemma gödel_spec : T₀ ⊢ (gödel B) ↔ ~B (gödel B)
```

補題 11 (Abstract G1, G2, Löb): T が無矛盾かつ対角化可能で, $\mathfrak{B}$ は HBL を満たすとする.

G1 $T \not\vdash G_{\mathfrak{B}}$. さらに $T \vdash \mathfrak{B} \sigma \implies T \vdash \sigma$ を満たすなら¹, $T \not\vdash \neg G_{\mathfrak{B}}$.

G2 $T \not\vdash \text{Con}_{\mathfrak{B}}$

Löb $T \vdash \mathfrak{B} \sigma \rightarrow \sigma$ なら $T \vdash \sigma$

F-Löb $T \vdash \mathfrak{B}(\mathfrak{B} \sigma \rightarrow \sigma) \rightarrow \mathfrak{B} \sigma$

```
class Kreisel [L.ReferenceableBy L] {T₀ T : Theory L} (B : Provability T₀ T) where
  KR {σ : Sentence L} : T ⊢ B σ → T ⊢ σ
```

```
theorem unprovable_gödel : T ⊄ (gödel B)
theorem unrefutable_gödel [B.Kreisel] : T ⊄ ~(gödel B)
theorem con_unprovable [Consistent T] : T ⊄ B.con
theorem löb_theorem (H : T ⊢ B σ → σ) : T ⊢ σ
theorem formalized_löb_theorem : T₀ ⊢ B (B σ → σ) → B σ
```

Source: [1](#)

¹この条件は Σ_1 -健全性を純粋に構文論的に仮定したものを見ることが出来る.

算術化を頑張ると HBL を満たす証明可能性を実際に構成することができる.

- 今後この標準的な証明可能性は $\Box_T$ と書く.

また, 対角化可能性も実際満たす. 故に系として, 第 2 不完全性定理や Löb の定理を機械化出来る.

定理 12 (Gödel の第 2 不完全性定理): T が $\text{I}\Sigma_1$ より強く, Δ_1 -定義可能で, 無矛盾なら, T の無矛盾性を表す文 $\neg\Box_T\perp$ は証明できない.

```
theorem consistent_unprovable [Consistent T] : T  $\not\vdash$  T.consistent.val
```

Source: [1](#)

定理 13 (Löb の定理): $T \vdash \Box_T \sigma \rightarrow \sigma$ なら $T \vdash \sigma$

```
theorem löb_theorem : T ⊢ provabilityPred T σ → σ → T ⊢ σ
```

Source: [1](#)

T として Peano 算術 PA を取ることが出来る。故に PA は不完全だし、その無矛盾性は PA 自身で証明できない。

```
instance : PA ≰ PA ∪ PA.Con  
instance : PA ≰ PA ∪ PA.Incon
```

$\mathfrak{B}$ としてこのような性質を考えよう

定義 14: $\text{Ros}: T \vdash \neg \sigma \implies T \vdash \neg \mathfrak{B} \sigma$

```
class Rosser [L.ReferenceableBy L₀] {T₀ : Theory L₀} {T : Theory L} (B : Provability T₀ T) where
  Ros {σ : Sentence L} : T ⊢ ~σ → T₀ ⊢ ~B σ
```

Source: [1](#)

定理 15 (Abstract GR): T が無矛盾かつ対角化可能で $\mathbb{B}$ が Ros を満たすなら, $T \not\vdash G_{\mathbb{B}}$ かつ, 何も仮定することなく $T \not\vdash \neg G_{\mathbb{B}}$.

```
local notation "R"  $\Rightarrow$  gödel  $\mathbb{B}$ 
```

```
theorem unrefutable_rosser [ $\mathbb{B}$ .Rosser] :  $T \not\vdash \sim R$ 
```

```
theorem rosser_independent [L.DecidableEq] [ $\mathbb{B}$ .Rosser] : Independent  $T$   $R$ 
```

```
theorem rosser_first_incompleteness [L.DecidableEq] ( $\mathbb{B}$  : Provability  $T_0$   $T$ ) [ $\mathbb{B}$ .Rosser] :  
  Incomplete  $T$ 
```

Source: [1](#)

これは Gödel-Rosser の第 1 不完全性定理の抽象的な主張となっている (Σ_1 -健全性を消去)

実際 witness comparison などの道具を使って Ros を満たす証明可能性述語を具体的に構成することが出来る。ゆえに

定理 16 (Gödel-Rosser の第 1 不完全性定理): T が $I\Sigma_1$ を含み, Δ_1 -定義可能で, 無矛盾なら, T から証明も反証も出来ない文が存在する.

```
theorem incomplete_GR (T : ArithmeticTheory) [T. $\Delta_1$ ] [ $I\Sigma_1 \leq T$ ] [Consistent T] : Incomplete T
```

Source: [1](#)

不完全性定理において無矛盾性を表す文は「 $\perp$ が証明されない」以外にもある.

- 例: 「任意の文は証明されかつ同時に反証されるということはない」

定理 17 (Jeroslow の第 2 不完全性定理): T が無矛盾なら, $T \not\vdash \forall x. \neg(\Box_T x \wedge \Box_T (\neg x))$

```
theorem unprovable_formalized_law_of_noncontradiction [IS1 ≤ T] [Entailment.Consistent T]
  : T ⊈ ∀1 ~ (provable T ∧ T.refutable)
```

Source: [1](#)

論理式 φ の Gödel 数「 φ 」を受け取って $\neg\varphi$ の Gödel 数「 $\neg\varphi$ 」を返す関数 $\neg x$ を考える必要があるが, 抽象化ではそれは面倒.

- 反証可能性 $\mathfrak{W}$ を考える. つまり, $T \vdash \neg\sigma \implies T \vdash \mathfrak{W}\sigma$ を考えれば良い.

```
structure Refutability [L.ReferenceableBy L0] (T0 : Theory L0) (T : Theory L) where
  refu : Semisentence L0 1
  refu_def {σ : Sentence L} : T ⊢ ~σ → T0 ⊢ refu / [「σ」]
```

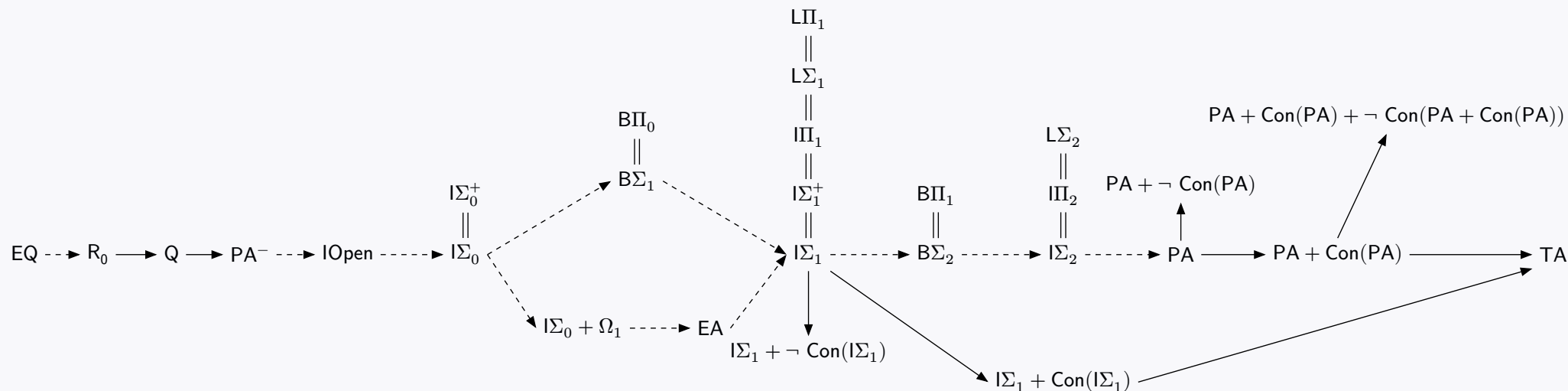
Source: [1](#)

各詳細は Saitou and Noguchi (2026) を読んでください (いくつかは次の版で追加予定).

- Tarski の真理定義不可能性定理.
- Church の一階述語論理の決定不可能性.
- 「Gödel 数が n 以下の証明で証明できる」証明可能性述語について¹.
- Ehrenfeucht-Mycielski の加速定理.
- Craig のトリック.
- Friedman–Goldfarb–Harrington 定理 (FGH 定理)².
- 第 1 不完全性定理が成立する理論の Lindenbaum 代数は全て同型.
- $I\Sigma_n: n \geq 1$ は有限公理化可能.
- PA は有限公理化不能 (Ryll-Nardzewski の定理).

¹この否定の不動点「Gödel 数が n 未満の証明では証明できない」は $\mathbb{N}$ 上で正しいし、実際に n 未満の証明で証明できない. 故に n を途方もなく大きく取ればそれは正しいが現実的な証明を持たない命題とも言える.

² $I\Sigma_1 + \text{Con}(T)$ 上では任意の Σ_1 -文は何らかの σ に対し $\Box_T \sigma$ と同値.



全部 strict だがそこまで出来てない... ¹

¹簡単な説明：

- $I\Sigma_1^+$ は今回の緩い Σ_1 論理式に対しての帰納法原理を追加した体系.
- $B\Sigma_1$ は Σ_1 -論理式に対しての採集原理.
- $L\Sigma_1$ は Σ_1 -論理式に対しての最小値原理.

4. 証明可能性論理

証明可能性論理の簡単な説明・モチベーション¹

- 不完全性定理において中心的な役割を果たす証明可能性述語「 φ は T で証明できる」を様相だと思おう.
- 証明可能性 $\Box$ をより様相論理的に扱う.

最重要の定理: Solovay の算術的完全性定理.

命題 (Solovay の算術的完全性定理(ラフ)): HBL な $\Box$ の挙動は様相命題論理 GL で完全に特徴づけられる.

```
theorem arithmetical_completeness_iff [T.SoundOnHierarchy  $\Sigma$  1] :  
  GL  $\vdash$  A  $\leftrightarrow$   $\forall f$  : Realization  $\alpha$   $\mathcal{L}_{or}$ ,  $T \vdash f$  T A
```

```
theorem eq_provabilityLogic [T.SoundOnHierarchy  $\Sigma$  1] : GL = T.provabilityLogic ( $\alpha$  :=  $\alpha$ )
```

Source: [1](#)

¹証明可能性論理の標準的な文献として Artemov and Beklemishev (2005), Boolos (1994), Smoryński (1985), Verbrugge (2024).

論理式は $\perp, \rightarrow, \Box$ を原始的な記号として導入し、あとは略記.

論理式の集合を論理と呼ぶ.

```
inductive Formula (α : Type*) where
  | atom    : α → Formula α
  | falsum  : Formula α
  | imp     : Formula α → Formula α → Formula α
  | box     : Formula α → Formula α
  deriving DecidableEq

abbrev Logic (α : Type*) := Set (Formula α)
```

Source: [1](#) [2](#)

α として Nat を取れば可算無限個の命題変数, Empty を取れば命題変数がない様相論理式 (閉論理式) を表せる.

定義 18: 論理 GL は古典命題論理に以下の公理と規則を足したもの.

- 規則 (Nec) : $\vdash A \implies \vdash \Box A$
- 公理 K : $\Box(A \rightarrow B) \rightarrow \Box A \rightarrow \Box B$
- 公理 4 : $\Box A \rightarrow \Box \Box A$
- 公理 L : $\Box(\Box A \rightarrow A) \rightarrow \Box A$

証明可能性 $\Box$ の HBL および F-Löb と以下で対応する.

- 規則 (Nec) が D1
- 公理 K が D2
- 公理 4 が D3
- 公理 L が F-Löb

```

inductive Logic.normalOf (A : Set (Formula α)) : Logic α
| axm {A}      : A ∈ A → normalOf A A
| mdp {A B}    : normalOf A (A → B) → normalOf A A → normalOf A B
| nec {A}      : normalOf A A → normalOf A (□A)
| verum        : normalOf A Axioms.Verum
| implyK {A B} : normalOf A (Axioms.ImplyK A B)
| implyS {A B C} : normalOf A (Axioms.ImplyS A B C)
| ...
| dne {A}      : normalOf A (Axioms.DNE A)
| axiomK {A B} : normalOf A (□(A → B) → □A → □B)

```

```

abbrev Logic.GL {α : Type*} : Logic α :=
  normalOf ({□A → □□A | A} ∪ {□(□A → A) → □A | A})

```

実用上はフレームのみを考えることは殆どないので、モデルだけを考えてほうが実装がスッキリする.

定義 19 (Kripke モデル): 非空集合 W とその上の 2 項関係 $\prec: W \times W \rightarrow 2$, 付値関数 $V: W \times \text{Prop} \rightarrow 2$ の組 $\langle W, \prec, V \rangle$ を Kripke モデルという. 強制関係(Forces)を以下で定める.

- $x \Vdash p \iff xVp$.
- $x \Vdash \Box A \iff \forall y, x \prec y \rightarrow y \Vdash A$.

Lean では, 型 κ 上の構造として定義する.¹

¹universe を自由にすることが出来ることは完全性定理などでそれなりに問題になるが, 無視する.

```

structure Model (κ : Type*) [Nonempty κ] (α : Type*) where
  Rel' : κ → κ → Prop
  Val' : κ → α → Prop

abbrev World (κ : Model κ α) := κ

abbrev Rel {M : Model κ α} : M.World → M.World → Prop := M.Rel'
scoped infix:60 " < " ⇒ Rel

abbrev Val {M : Model κ α} : M.World → α → Prop := M.Val'

def Forces (M : Model κ α) (x : M.World) : Formula α → Prop
| #a      ⇒ M x a
| ⊥       ⇒ False
| A → B   ⇒ Forces M x A → Forces M x B
| □A      ⇒ ∀ y, x < y → Forces M y A

scoped notation:55 x:56 " ⊨ [" M "]" " A:56 ⇒ Forces M x A

```

定義 20:

- **GL-モデル**とは $\prec$ が推移的で逆整礎的: $x_1 \prec x_2 \prec \dots \prec x_n$ が有限の n 回遷移しか出来ないとする.
 - 故に GL-モデルには点 x から最大何回遷移できるか: **ランク rank(x)** が定まる¹.
- **有限 GL-モデル**とは W が有限で $\prec$ が推移的で非反射的なモデルとする.
- モデルが**根付き**とは根 r_M があって任意の $x \in M \setminus \{r_M\}$ に対し $r_M \prec x$.
 - 根付きモデルの高さとは $\text{rank}(r_M)$ とする.

¹技術的な面倒のため, Lean での実装では有限モデルのみに対して定めている.

Lean では次のように定義する.

```
class IsGL (M : Model κ α) extends IsTrans _ M.Rel, IsConverseWellFounded _ M.Rel

class IsFiniteGL (M : Model κ α) extends IsTrans _ M.Rel, Std.Irrefl M.Rel where
  [finite : Finite M.World]

noncomputable def World.rank (x : M.World) : ℕ := cwfHeight (· < ·) x

structure RootedModel (κ : Type*) [Nonempty κ] (α : Type*) extends Model κ α where
  root : toModel.World
  root_rel : ∀ x, x ≠ root → root < x

noncomputable def RootedModel.height (M : RootedModel κ α) [Fintype M.World] [M.IsGL] : ℕ :=
  Model.World.rank (M := M.toModel) M.root
```

Source: [1](#) [2](#) [3](#)

余談:

もちろん様相論理一般の議論ではモデルとフレームの区別は非常に重要なのだが、今回は一般論をしたいわけではないので無視する.

- あくまでも算術を分析するための道具として割り切って実装する.
- 定理証明支援系で何を実装して何を実装しないか...
- **道具としての様相論理！**

Sambin and Valentini (1982) の GL のシークエント計算を機械化する. Kripke 意味論に対する完全性の証明はこちらのほうがはるかにスッキリする¹.

定義 21: シークエント $\Gamma \Rightarrow \Delta$ とは論理式の有限集合の組である. GL のシークエント計算 $\mathcal{G}_{GL}$ は次の規則からなる. ただし (WL) と (WR) では $\Gamma \subseteq \Gamma'$, $\Delta \subseteq \Delta'$ とする.

$$\begin{array}{c}
 \frac{}{A \Rightarrow A} (\text{Ax}) \qquad \frac{}{\perp \Rightarrow} (\perp\text{L}) \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma' \Rightarrow \Delta} (\text{WL}) \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta'} (\text{WR}) \\
 \\
 \frac{\Gamma \Rightarrow A, \Delta \quad B, \Gamma \Rightarrow \Delta}{A \rightarrow B, \Gamma \Rightarrow \Delta} (\rightarrow\text{L}) \qquad \frac{A, \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \rightarrow B, \Delta} (\rightarrow\text{R}) \qquad \frac{\Box A, \Gamma, \Box \Gamma \Rightarrow A}{\Box \Gamma \Rightarrow \Box A} (\Box_{GL})
 \end{array}$$

¹GL は標準的な様相論理のカノニカルモデルの構成による証明は不可能. 有限反例モデルを直接作るか, 適当にモデルを同値で割って証明する. Hilbert 流だと細かい部分の計算が煩雑になる.

```

structure Sequent (α : Type*) where
  ant : FormulaFinset α
  suc : FormulaFinset α

infix:50 " ⇒ " => Sequent.mk

inductive Gentzen : Sequent α → Prop
| axm (A) : Gentzen ({A} ⇒ {A})
| botL : Gentzen ({⊥} ⇒ ∅)
| wkL {Γ Γ' Δ} : Gentzen (Γ ⇒ Δ) → ( _ : Γ ⊆ Γ' := by grind ) → Gentzen (Γ' ⇒ Δ)
| wkR {Γ Δ Δ'} : Gentzen (Γ ⇒ Δ) → ( _ : Δ ⊆ Δ' := by grind ) → Gentzen (Γ ⇒ Δ')
| impL {Γ Δ A B} :
  Gentzen (Γ ⇒ insert A Δ) → Gentzen (insert B Γ ⇒ Δ) → Gentzen (insert (A → B) Γ ⇒ Δ)
| impR {Γ Δ A B} : Gentzen (insert A Γ ⇒ insert B Δ) → Gentzen (Γ ⇒ insert (A → B) Δ)
| boxGL {Γ A} : Gentzen (insert (□A) (Γ ∪ Γ.box) ⇒ {A}) → Gentzen (Γ.box ⇒ {□A})

notation:45 "⊢G[GL] " S:50 => Gentzen S

```

定理 22 ($\mathcal{G}_{GL}$ の完全性定理): $\mathcal{G}_{GL} \vdash \Gamma \Rightarrow \Delta$ と任意の有限 GL-モデル M で $M \models \bigwedge \Gamma \rightarrow \bigvee \Delta$ であることは同値.

```
def Model.World.ForcesSequent (M : Model κ α) (x : M.World) (S : Sequent α) : Prop :=
  (∀ C ∈ S.ant, x ⊨ C) → ∃ D ∈ S.suc, x ⊨ D

def Model.ValidateSequent (M : Model κ α) (S : Sequent α) : Prop := ∀ x : M.World, x ⊨ S

lemma GL.Gentzen.iff_valid : ⊢G[GL] S ↔
  ∀ {κ : Type u} [Nonempty κ] (M : Kripke.Model κ α), [M.IsFiniteGL] → M ⊨ S
```

Source: [1](#) [2](#)

完全性定理から、カット規則の許容性（意味論的カット除去）がすぐに従う。

定理 23 ($\mathcal{G}_{GL}$ のカット除去定理): カット規則は $\mathcal{G}_{GL}$ で許容される。

$$\frac{\Gamma_1 \Rightarrow \Delta_1, A \quad A, \Gamma_2 \Rightarrow \Delta_2}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2} (\text{Cut})$$

`theorem GL.Gentzen.cut` ($h_1 : \vdash^G[GL] \Gamma_1 \Rightarrow \text{insert } A \Delta_1$) ($h_2 : \vdash^G[GL] \text{insert } A \Gamma_2 \Rightarrow \Delta_2$) :
 $\vdash^G[GL] \Gamma_1 \cup \Gamma_2 \Rightarrow \Delta_1 \cup \Delta_2$

Source: [1](#)

あとは頑張れば Hilbert 流と対応することがわかる．故に次の同値が言える．

定理 24: 以下同値． A は論理式．

1. $GL \vdash A$
2. $\mathcal{G}_{GL} \vdash \Rightarrow A$
3. 任意の有限 GL-モデルで A は妥当．
4. 任意の有限 GL-モデルの根で A は充足される．
5. 任意の有限 GL-木モデルの根で A は充足される．

```
theorem Logic.GL.provability_TFAE : [
  GL ⊢ A,
  ⊢G[GL] ∅ ⇒ {A},
  ∀ {κ : Type u} [Nonempty κ] (M : Model κ α), [M.IsFiniteGL] → M ⊨ A,
  ∀ {κ : Type u} [Nonempty κ] (M : RootedModel κ α), [M.IsFiniteGL] → M.root ⊨ A,
  ∀ {κ : Type u} [Nonempty κ] (M : RootedModel κ α), [M.IsFiniteGL] → [M.IsTree] →
    M.root ⊨ A
].TFAE
```

今回は我々の関心外であるためカット除去は意味論的に行ったが、構文論的なカット除去も可能である。

シーケントが有限集合ではなくリストや多重集合である場合の構文論的なカット除去は本当に帰納法が回っていたのか最近まで不明であった。(Goré and Ramanayake (2012) で **3 重帰納法**によって示される。) Brighton (2016) が新しい方法を提案していて、それがうまくいくということは Goré et al. (2021) が Rocq で検証している。

シーケントを集合で定義しているため、例えば weakening や contraction をどのタイミングで行うかは決定的ではなく、**(Lean において) 現実的に証明探索は出来ない¹**。

例えば、Maggesi and Perini Brogi (2023) はラベル付きシーケント計算体系を HOL Light で実装し、GL_provable のような論理式が GL で証明できるか判定するタクティクを定義している。

- ただし、このタクティクが必ず停止することは HOL Light の中では証明されていない。数学的に停止するというメタの保証 Negri (2005) に依存して実装されている。(つまり実装が間違っていたら永遠に止まらないという可能性はある。)
- 一応我々はラベル付きシーケント計算も実装して停止性も機械化したけど...

¹多分多重集合であったとしても Lean では決定的に証明できないと思う。リストなら出来る気もするが、あまりにも面倒な実装になると思う。

そもそも機械化の上では命題変数 p, q として $p \rightarrow \Box q$ ではなく, メタの論理式変数 A, B として $A \rightarrow \Box B$ は証明可能か? と要請されることがほとんどのため, これらのタクティクはそこまで役に立たなかった.

実用的には, Kripke 意味論で一旦健全・完全性を使って証明出来る・出来ないを示せば良く, それでも別にある程度自動化できたりするので十分.

Sambin and Valentini (1982) ではさらにシークエント計算を使った応用として, GL の Craig 補間定理と不動点定理を示している. GL の導出木を手作りすれば補間と不動点は構成的に計算できる (Maehara の方法)¹².

定理 25 (GL の Craig 補間性): $GL \vdash A \rightarrow B$ ならば, 論理式 C で $GL \vdash A \rightarrow C$ かつ $GL \vdash C \rightarrow B$ であり, C の命題変数がすべて A と B の両方に現れるものが存在する.

```
theorem Logic.GL.CIP (h : GL ⊢ A → B) :  
  ∃ C, GL ⊢ A → C ∧ GL ⊢ C → B ∧ C.atoms ⊆ A.atoms ∩ B.atoms
```

Source: [1](#) [2](#)

¹ただし殆どの場合そんなことはなくて完全性から作るので意義はない.

²Gignoux (2026) も Lean で補間性定理などを示しているが, 意味論的な制約により特殊なケースのみになっている. シークエント計算から一般に構成出来るというのはそれなりに利点に思える.

重要な系として, GL の補間定理から不動点定理も従う.

定義 26: 命題変数 p が論理式 A で **modalized** であるとは, A に現れる p がすべて $\Box$ のスコープの中にあることをいう.

```
def ModalizedIn (p :  $\alpha$ ) : Formula  $\alpha$   $\rightarrow$  Prop
| #a       $\Rightarrow$  a  $\neq$  p
|  $\perp$       $\Rightarrow$  True
| A  $\rightarrow$  B  $\Rightarrow$  A.ModalizedIn p  $\wedge$  B.ModalizedIn p
|  $\Box$  _     $\Rightarrow$  True
```

Source: [1](#)

定理 27 (GL の不動点定理 Sambin and Valentini (1982)): p が A で modalized ならば, p を含まず A の命題変数のみからなる論理式 D で

$$\text{GL} \vdash A[p := D] \leftrightarrow D$$

を満たすものが存在する. さらに不動点は証明可能同値を除いて一意である: $\text{GL} \vdash A[p := E] \leftrightarrow E$ なる任意の論理式 E について $\text{GL} \vdash D \leftrightarrow E$.

```
theorem Logic.GL.fixpoint_theorem (hpq : p ≠ q) (hA : A.ModalizedIn p) (hq : q ∉ A.atoms) :
  ∃ D, D.atoms ⊆ A.atoms.erase p ∧ GL ⊢ A[p ↦ D] ↔ D ∧
  ∀ E, GL ⊢ A[p ↦ E] ↔ E → GL ⊢ D ↔ E
```

GL の $\Box$ と証明可能性 $\mathfrak{B}$ を結びつけよう.

定義 28: f を様相論理の命題変数から算術の文への写像とし **実現(realization)** と呼ぶ. f による様相論理式 A の **$\mathfrak{B}$ -解釈 $f_{\mathfrak{B}}(A)$** を以下で定める.

- $f_{\mathfrak{B}}(p) = f(p)$
- $f_{\mathfrak{B}}(\perp) = \perp$
- $f_{\mathfrak{B}}(A \rightarrow B) = f_{\mathfrak{B}}(A) \rightarrow f_{\mathfrak{B}}(B)$
- $f_{\mathfrak{B}}(\Box A) = \mathfrak{B} f_{\mathfrak{B}}(A)$

ただし今後は標準的な証明可能性述語 $\Box_T$ だけで議論するので $f_{\Box_T}(A)$ のみ考える.

```
structure Realization (α : Type*) (L : Language) where
  val : α → Sentence L

def interpret (f : Realization α L) (ℬ : Provability T₀ T) : Formula α → Sentence L
| #a      ⇒ f.val a
| ⊥       ⇒ ⊥
| A → B   ⇒ A.interpret f ℬ → B.interpret f ℬ
| □A      ⇒ ℬ (A.interpret f ℬ)

noncomputable abbrev standardInterpret (f : Realization α Lₒᵣ)
  (T : ArithmeticTheory) [T.Δ₁] : Formula α → Sentence Lₒᵣ :=
  interpret f T.standardProvability
```

定義 29: 算術 T, U とする. U 上の T の証明可能性論理 $\text{PL}_T(U)$ を

$$\text{PL}_T(U) := \{ A \text{ は様相論理式} \mid \text{任意の実現 } f \text{ に対し } U \vdash f_{\Box_T}(A) \}$$

で定める.

```
def ArithmeticTheory.provabilityLogicRelativeTo
  (T U : ArithmeticTheory) [T.Δ₁] : Logic α :=
  { A | ∀ f : Realization α Lor, U ⊢ f T A }

abbrev ArithmeticTheory.provabilityLogic (T : ArithmeticTheory) [T.Δ₁] :
  Logic α :=
  T.provabilityLogicRelativeTo T
```

定理 30 (Solovay の算術的完全性定理(Solovay (1976))): $PL_{PA}(PA) = GL$.

```
theorem arithmetical_completeness_iff [T.SoundOnHierarchy Σ 1] :  
  GL ⊢ A ↔ ∀ f : Realization α Lor, T ⊢ f T A
```

```
theorem eq_provabilityLogic [T.SoundOnHierarchy Σ 1] : GL = T.provabilityLogic (α := α)
```

Source: [1](#)

証明: 健全性 $\Leftarrow$ は HBL と F-Löb より簡単にわかる. 完全性 $\Rightarrow$ が難しい. 対偶をと
り, 反例 GL-木モデルを算術の中に埋め込んで満たさない実現を構成する. ■

$PL_T(U)$ の T, U を動かすとどうなるかは Beklemishev (1990) によって完全に分類されている.

定義 31: 以下の論理を定める. $GL + X$ は X との union の MP/subst の閉包 (非正規拡大).

- $GL_\alpha(X) := GL + \{\Box^{n+1}\perp \rightarrow \Box^n\perp : n \in X\}.$
- $GL_\beta(X) := GL + \neg \bigwedge_{n \notin X} (\Box^{n+1}\perp \rightarrow \Box^n\perp) : \text{ただし } X \text{ は補有限}.$
- $A := GL + \{\neg\Box^n\perp : n \in \mathbb{N}\}.$
- $D := GL + \neg\Box\perp + \Box(\Box A \vee \Box B) \rightarrow \Box A \vee \Box B.$
- $S := GL + \Box A \rightarrow A.$

4.10 証明可能性論理の分類定理

```

inductive Logic.sumQuasiNormal (L1 L2 : Logic α) : Logic α
| mem1 {A}      : A ∈ L1 → sumQuasiNormal L1 L2 A
| mem2 {A}      : A ∈ L2 → sumQuasiNormal L1 L2 A
| mdp {A B}      :
    sumQuasiNormal L1 L2 (A → B) → sumQuasiNormal L1 L2 A → sumQuasiNormal L1 L2 B
| subst {A s}    : sumQuasiNormal L1 L2 A → sumQuasiNormal L1 L2 (A[s])

infix:50 " +L " ⇒ Logic.sumQuasiNormal

abbrev Logic.S {α : Type*} : Logic α := GL +L { □A → A | A }
notation "S" ⇒ Logic.S

```

定義 32: 論理式 A のトレース

$\text{tr}(A) := \{n \in \mathbb{N} : r_M \Vdash A \text{ となる高さ } n \text{ の有限根付きモデル } M \text{ が存在}\}$

論理 L のトレース $\text{tr}(L) := \bigcup_{A \in L} \text{tr}(A)$.

```
def Formula.trace (A : Formula α) : Set ℕ :=
  {n | ∃ (κ : Type u) ( _ : Nonempty κ) (M : RootedModel κ α)
    ( _ : Fintype M.World) ( _ : M.IsGL), M.height = n ∧ M.root ⊭ A}

def Logic.trace (L : Logic α) : Set ℕ := ⋃ A ∈ L, A.trace
```

定義 33: $T \vdash \Box_T^n \perp$ となる最小の n を **理論 T の高さ $\text{hgt}(T)$** という。なければ ω 。

```
noncomputable def Provability.height (B : Provability T₀ T) : ENat := ENat.find (T ⊢ B^[.] ⊥)
```

```
lemma height_eq_top_iff : B.height = τ ↔ ∀ n, T ⊄ B^[n] ⊥
```

```
noncomputable abbrev ArithmeticTheory.height (T : ArithmeticTheory) [T.Δ₁] : ℕ∞ :=  
  T.standardProvability.height
```

Source: [1](#)

定理 34 (Beklemishev (1990)): T を $I\Sigma_1$ を含む Δ_1 -定義可能な算術, U を任意の算術とし, $L := \text{PL}_T(U)$, $X := \text{tr}(L)$ とする.

1. X が補無限なら $L = \text{GL}_\alpha(X)$.
2. X が補有限かつ $L \not\subseteq S$ なら $L = \text{GL}_\beta(X)$.
3. X が補有限かつ $L \subseteq S$ なら L は $\text{GL}_\alpha(X)$, $D \cap \text{GL}_\beta(X)$, $S \cap \text{GL}_\beta(X)$ のいずれか.

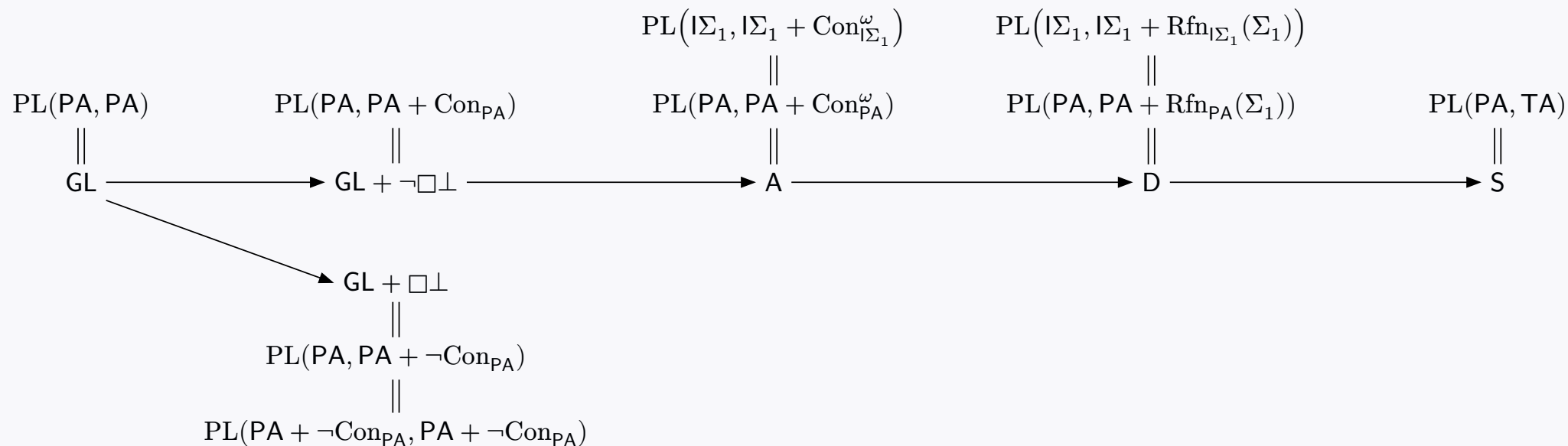
```
theorem provabilityLogic_classification :
  letI L := T.provabilityLogicRelativeTo U (α := α);
  L = GLα L.trace v
  ∃ hL : L.traceᶜ.Finite, L = GLβ _ hL v L = D ∩ GLβ _ hL v L = S ∩ GLβ _ hL
```

定理 35 (Beklemishev (1990)): T を $\text{I}\Sigma_1$ を含む Δ_1 -定義可能な算術とし, 真の算術の証明可能性論理 $L := \text{PL}_T(\text{TA})$ について.

1. T が健全なら $L = \text{S}$.
2. 健全ではないが Σ_1 -健全なら $L = \text{D}$.
3. Σ_1 -健全でなく $\text{hgt}(T) = \omega$ なら $L = \text{A}$.
4. $\text{hgt}(T) = n < \omega$ なら $L = \text{GL}_\beta(\mathbb{N} \setminus \{n\})$.

のいずれか一つのみが成立する.

```
theorem provabilityLogic_TA_classification : [
  NDown[L_Or] ⊢* T ∧ T.provabilityLogicRelativeTo TA (α := α) = S,
  T.SoundOnHierarchy Σ 1 ∧ ¬NDown[L_Or] ⊢* T ∧ T.provabilityLogicRelativeTo TA (α := α) = D,
  ¬T.SoundOnHierarchy Σ 1 ∧ T.height = τ ∧ T.provabilityLogicRelativeTo TA (α := α) = A,
  ∃ n : ℕ, T.height = n ∧ T.provabilityLogicRelativeTo TA (α := α) = GLβ {n}^c (by simp)
].OA00
```



原理的には，それぞれの間には 2^ω 個の相異なる証明可能性論理がある（はず）。

5. まとめと今後の展望と余談

まだまだ山程の課題がある！

- 算術的階層をどう扱う？
- 理論の解釈(interpretation)の実装
- 部分真理定義述語とそれを用いた議論
- もっと弱い算術
- PA などの独立命題 (Paris-Harrington の定理など)
- 直観主義(述語論理・算術・様相論理)
- 集合論

などなど...

Heyting 算術 HA（直観主義論理上の PA）の証明可能性論理は 50 年ぐらいの問題¹.

- 最近の進展として Mojtabedi (2026) が完全な公理化を発表したが、数年間は査読中².
- **これを機械化して厳密に検証したい.**

直観主義様相論理：

- iGL：直観主義命題論理 + K + (Nec) + $\Box(\Box A \rightarrow A) \rightarrow \Box A$. HA に対して算術的に健全 (Litak (2014), van der Giessen and Iemhoff (2021)).
- iSL：iK + $(\Box A \rightarrow A) \rightarrow A$ (Visser and Litak (2024)).

Rocq での先行研究（様相論理的な興味）

- iGL のカット除去 Goré and Shillito (2022), Shillito (2022)
- iSL のカット除去 Shillito et al. (2023), 一様補間定理 Férée et al. (2024).

¹Artemov and Beklemishev (2005), Beklemishev and Visser (2006)

²v1 は 2022 年出版

Heyting 算術をまずどう形式化するか？

- HA の証明をどうやって書く？
- 完全性経由で一旦モデルに持ってくるみたいな議論は出来なさそう.

解決案の候補：Lean の中で HA のメタ定理証明支援系を作って形式証明を書く？

```
example : HA proves " $\forall x, \forall y, 0 < x + y + 1$ " := work_in
  HA_intro;
  ...
```

公理的集合論の形式化はたまにやられている.

- Lean3 で CH の独立性 (**flyspeck プロジェクト**: Han and van Doorn (2020)).
 - ▶ ただし Boole 値モデルによるもので, 強制法より適用範囲が狭いという問題点がある.
- Isabelle/ZF では, 強制法による AC, $\neg$ AC や CH の独立性が形式化されている cf: Gunther et al. (2020), Gunther et al. (2022), Paulson (2003), 舟根 (2024).

ただしこれ以上の進展はあまりない (様に見える).

強制法の一般的な枠組みを機械化して、連続体仮説の独立性などを示したい.

- 方針 1: 可算推移的モデル M と generic filter G から $M[G]$ を作る標準的な方法 (cf: Kunen (2011)).
- 方針 2: 証明論的強制法 (cf: Avigad (2004)): 理論間の強制解釈と保存性を示す. ZFC の無矛盾性だけから独立性が従い, 証明の変換も有限的.
 - 一階述語論理の完全性定理の機械化では既に簡略化された強制法を使っている.

その他の問題: 半順序などに関する $\text{Lean} \vdash \varphi$ の議論をどう $\text{Lean} \vdash \text{"ZFC} \vdash \varphi\text{"}$ へどうやって持っていく?¹

¹Lean の型理論から考えるとこれを素朴に変換する仕組みが作れるとは思えない. フルスクラッチで書き直す必要もある?

考えたいこと？

- もっと集合論の独立命題.
- 巨大基数の比較：<https://neugierde.github.io/cantors-attic/>.
- 強制法の様相論理：強制法の原理は様相論理 S4.2 の拡張体系でおおよそ記述出来る.

現状, Formalized Formal Logic の今回の形式化は Mathlib へ移植する気はほとんどない.

OSS 開発としての問題

- Mathlib のレビューは遅く, また論理学の適切なレビュアーが今いない.
- Mathlib の肥大化: ライブラリが 8GB もあるのどうなの?

論理学特有の問題

- 論理学はどのような言語や定式化を選択するかなどの定義に強く依存する.
 - そのようなものを汎用的なライブラリとして提供する (数学的・技術的) 方法がない, あるいは意義が薄い.

今後はあくまでも FFL のフレームワークで論理学をやる方面になるだろう (デファクト・スタンダードになればいいな)

Formal Conjectures¹ という Lean による未解決問題のカタログがある. これの FFL 版を作ろう.

例えば算術には以下の問題がある

1. $I\Delta_0$ は有限公理化可能か?
2. $I\Delta_0$ で素数が無限であることは証明可能か?

このような問題をカタログ化しておけば誰か (何か) が取り組んでくれるかも?²

¹<https://github.com/google-deepmind/formal-conjectures>

²それが果たしてどうなんだ? という問題はありますが.

注意 (再掲):

1. 依拠している数学的基礎に問題や矛盾が無いのか? (論理学者として)
2. ソフトウェアとしての実装にバグが存在しないか? (ソフトウェア開発者として)
3. 形式化したそのコードは命題を正しく形式化しているか? (エンドユーザとして)

「定理証明支援系を開発する」という目的でなかったとしても、形式証明を書くという事業はソフトウェア開発の一種である。

OpenAI, Navier-Stokes 方程式に関する形式化のコミットの例.



(Anthropic の FLT の定理の形式化はまず GitHub のプレビューが追いついていなかった)

OpenAI や Anthropic などが提出した NS 方程式や FLT の形式証明は問題があると個人的には思う。

妥当性の問題 本当に正しく主張を形式化出来てる？

- 100k 行の証明をどうやって一気に人間がレビューしろと？
- この膨大なコードの中で定理証明支援系自体の実装のバグを踏んでいないとなぜ言える？

重要度の問題 アドホックな補題や定義・議論が多すぎるのでは？

- 1 回しか使わない補題など、本当に切り出すべきか？

再利用性の問題 車輪の再発明では？

- 難問にはそれなりにたくさんの道具が必要，それらを毎回毎回用意し直すのは無駄。
- 適切にライブラリとして切り出せばコミュニティとしても嬉しいはず。しかし一切そのようなことが行われている気配はない¹。

¹ただし現状では Mathlib は AI/LLM による PR は受け付けていない。それでは困るので、最近では Tau Ceti Project <https://github.com/TauCetiProject/TauCeti> という AI の使用やレビューを受け付けている実験場も用意されている。

定理証明支援系・形式証明を

- ・ 単なる自然言語による証明のより精緻な保証の執筆作業としてではなく,
- ・ 数学をより高速かつ正確に生産するソフトウェア開発の視点を投げかける: **証明工学**

一般のソフトウェア開発にはあまりない面白い課題が多く眠っていると思う。

deformalization 逆方向, つまり, 既にある形式証明から逆に自然言語で書かれた読みやすい証明へと翻訳する. あるいは, そもそもこの形式証明が書かれた意図などを補題から抽出することで, 新たな数学的理論の創出に繋がる可能性がある.

visualization 定理や補題同士の繋がりや, そもそも数学的構造自体の可視化を自動で行えたら, それは形式証明を人力で読み書きする人間にも役立つだろう.

collaborating 数学というかなり専門性や属人性の高い知識が要求される開発者をどのようにマネジメントしたりプロジェクトを進めるべきか?

architecture もっと効率的・正確に形式証明を生成するような自動証明アーキテクチャはどのようなものだろうか?

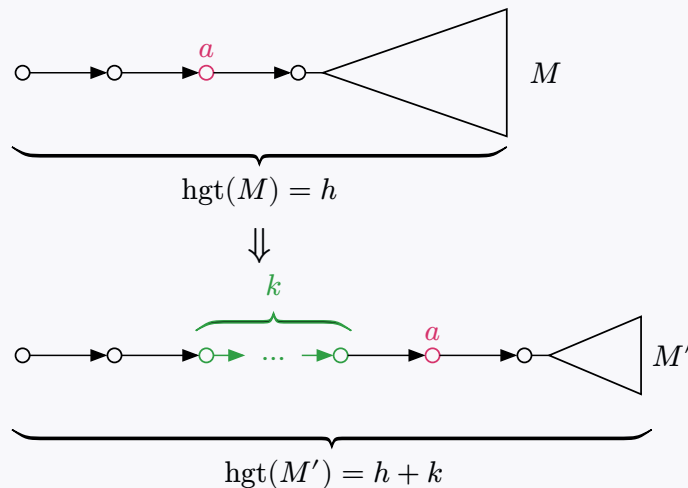
もちろん古典的なソフトウェア工学的な知見だって大いに役立つだろう¹.

¹ソフトウェア工学という学問が現代ないし現場のソフトウェア開発に果たしてどれだけ寄与したのか, あるいは単に机上の空論ではなかったのか? という批判はありますが.

Solovay の算術的完全性定理までは LLM は積極的に利用していなかった (2025 年ごろ).
積極的に利用し始めたのは証明可能性論理の分類定理を機械化するにあたって (2026 年 7 月).
技術的に面倒な部分があって停滞していたが, 大きく加速した.

Kripke モデルの幾何的な議論（**絵で書いたら自明じゃん**）をどう Lean でコードを書くべきか？が難しく（あるいはただ面倒な議論で）詰まっていた。

例: モデル M の適当な点 a の前に $k \leq \omega$ 個の点を追加する. このとき, 新しいモデル M' の高さ $\text{hgt}(M) + k$ は有限個の追加ならせいぜい $\text{hgt}(M) + k$.



機械化するとなると結構面倒な議論になる.

AI/LLM は Mathlib の全てのコードを学習しており(おそらく), 自分が知らなかったようなタクティクや方法の形式証明を書いてくれる.

個人的には, 来年(2027 年)にはもうそれほど人力で形式証明を書かなくても良い時代が来るのかなと思う.

- 便利なキーボードとしての AI/LLM.
- もちろん全体的な議論や実装の筋の良さ, みたいなものは人が評価するべきだと思う.

最近では AlphaCentauri というリポジトリで AI/LLM 生成した算術周りの証明を置いています.

- <https://github.com/FormalizedFormalLogic/AlphaCentauri>
- 人間半分, AI 半分で自動証明.
 - 人間の担当箇所: 証明の方針や議論全体の設計や切り分け. PR のレビュー.
 - AI の担当箇所: 人間の方針をもとに実際のコード量の見積もりや形式証明の執筆.

レビュー出来ていないが外部の協力者から PA の独立命題の形式化も送られてきている.

- <https://github.com/FormalizedFormalLogic/goodstein-independence>
- Goodstein 列の停止性, Hydra ゲーム, Paris-Harrington の定理など.

この発表では定理証明支援系 Lean で Gödel の不完全性定理と証明可能性論理に関する事実を形式化したということを発表しました.

定理証明支援系は楽しいので皆さんも触ってみてください.

- ・ 将来は人間のプレプリントを全部形式化して厳密に追試することが出来る, みたいな世界ができればそれは素晴らしいことだと思います.
- ・ ロジックの形式化をしている人や資産が**本当に**足りてないので, 知的資源およびソフトウェア的な資源で協力してくれる人がいたら本当に助かります.

説明出来なかった機械化の技術的な詳細は 齋藤 (2024) にも記載がある (それほど実装の本筋は変わっていない).

@Alwe_Logic, 2026. https://x.com/Alwe_Logic/status/2099488711427973188 [WWW Document].. URL https://x.com/Alwe_Logic/status/2099488711427973188

Artemov, S.N., Beklemishev, L.D., 2005. Provability Logic, in: Gabbay, D.M., Guenthner, F. (Eds.), Handbook of Philosophical Logic, 2nd Edition. Springer Netherlands, Dordrecht, pp. 189–360.. https://doi.org/10.1007/1-4020-3521-7_3

Avigad, J., 2004. Forcing in Proof Theory. The Bulletin of Symbolic Logic 10, 305–333.. <https://doi.org/10.2178/bsl/1102022660>

Avigad, J., 2001. Algebraic Proofs of Cut Elimination. The Journal of Logic and Algebraic Programming 49, 15–30.. [https://doi.org/10.1016/S1567-8326\(01\)00009-1](https://doi.org/10.1016/S1567-8326(01)00009-1)

Beklemishev, L.D., 1990. On the Classification of Propositional Provability Logics. Mathematics of the USSR-Izvestiya 35, 247–275.. <https://doi.org/10.1070/IM1990v035n02ABEH000701>

Beklemishev, L.D., Visser, A., 2006. Problems in the Logic of Provability, in: Gabbay, D.M., Goncharov, S.S., Zakharyashev, M. (Eds.), Mathematical Problems from Applied Logic I. Springer, New York, pp. 77–136.. https://doi.org/10.1007/0-387-31072-X_2

Boolos, G., 1994. The Logic of Provability. Cambridge University Press, Cambridge.. <https://doi.org/10.1017/CBO9780511625183>

Brighton, J., 2016. Cut Elimination for GLS Using the Terminability of Its Regress Process. Journal of Philosophical Logic 45, 147–153.. <https://doi.org/10.1007/s10992-015-9368-4>

Carneiro, M., 2019. The Type Theory of Lean (Master's thesis).

Férée, H., van der Giessen, I., van Gool, S., Shillito, I., 2024. Mechanised Uniform Interpolation for Modal Logics K, GL, and iSL, in: Benzmüller, C., Heule, M.J.H., Schmidt, R.A. (Eds.), Automated Reasoning. Springer Nature Switzerland, Cham, pp. 43–60..

https://doi.org/10.1007/978-3-031-63501-4_3

Gignoux, M., 2026. Proofs as Coalgebras (Master's thesis).

Goré, R., Ramanayake, R., 2012. Valentini's Cut-Elimination for Provability Logic Resolved. The Review of Symbolic Logic 5, 212–238.. <https://doi.org/10.1017/S1755020311000323>

Goré, R., Ramanayake, R., Shillito, I., 2021. Cut-Elimination for Provability Logic by Terminating Proof-Search: Formalised and Deconstructed Using Coq, in: Das, A., Negri, S. (Eds.), Automated Reasoning with Analytic Tableaux and Related Methods. Springer International Publishing, Cham, pp. 299–313.. https://doi.org/10.1007/978-3-030-86059-2_18

Goré, R., Shillito, I., 2022. Direct Elimination of Additive-Cuts in GL4ip: Verified and Extracted, in: Fernández-Duque, D., Palmigiano, A., Pinchinat, S. (Eds.), Advances in Modal Logic. College Publications, London, pp. 429–450.

Gunther, E., Pagano, M., Sánchez Terraf, P., 2020. Formalization of Forcing in Isabelle/ZF, in: Peltier, N., Sofronie-Stokkermans, V. (Eds.), Automated Reasoning. Springer International Publishing, Cham, pp. 221–235.. https://doi.org/10.1007/978-3-030-51054-1_13

Gunther, E., Pagano, M., Sánchez Terraf, P., Steinberg, M., 2022. The Independence of the Continuum Hypothesis in Isabelle/ZF [WWW Document].. URL https://isa-afp.org/entries/Independence_CH.html

Han, J.M., van Doorn, F., 2020. A Formal Proof of the Independence of the Continuum Hypothesis. Association for Computing Machinery, New Orleans, LA, USA, pp. 353–366.. <https://doi.org/10.1145/3372885.3373826>

Kunen, K., 2011. Set Theory. College Publications, London.

- Litak, T., 2014. Constructive Modalities with Provability Smack, in: Bezhanishvili, G. (Ed.), Leo Esakia on Duality in Modal and Intuitionistic Logics. Springer Netherlands, Dordrecht, pp. 187–216.. https://doi.org/10.1007/978-94-017-8860-1_8
- Maggesi, M., Perini Brogi, C., 2023. Mechanising Gödel–Löb Provability Logic in HOL Light. Journal of Automated Reasoning 67.. <https://doi.org/10.1007/s10817-023-09677-z>
- Mojtahedi, M., 2026. On the Provability Logic of HA.. <https://doi.org/10.48550/arXiv.2206.00445>
- Moura, L. de, 2026. Postmortem for Kernel Soundness Bug #14576.
- Negri, S., 2005. Proof Analysis in Modal Logic. Journal of Philosophical Logic 34, 507–544.. <https://doi.org/10.1007/s10992-005-2267-3>
- O'Connor, R., 2005. Essential Incompleteness of Arithmetic Verified by Coq, in: Hurd, J., Melham, T. (Eds.), Theorem Proving in Higher Order Logics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 245–260.. https://doi.org/10.1007/11541868_16
- Paulson, L.C., 2014. A Machine-Assisted Proof of Gödel's Incompleteness Theorems for the Theory of Hereditarily Finite Sets. The Review of Symbolic Logic 7, 484–498.. <https://doi.org/10.1017/S1755020314000112>
- Paulson, L.C., 2003. The Relative Consistency of the Axiom of Choice Mechanized Using Isabelle/ZF. LMS Journal of Computation and Mathematics 6, 198–248.. <https://doi.org/10.1112/S1461157000000449>
- Saitou, S., Noguchi, M., 2026. Mechanizing Gödel's Incompleteness Theorems and Provability Logic.
- Sambin, G., Valentini, S., 1982. The Modal Logic of Provability. The Sequential Approach. Journal of Philosophical Logic 11, 311–342.. <https://doi.org/10.1007/BF00293433>
- Shankar, N., 1986. Proof-Checking Metamathematics (Theorem-Proving) (Ph.D. thesis).
- Shillito, I., 2022. New Foundations for the Proof Theory of Bi-Intuitionistic and Provability Logics Mechanized in Coq (Ph.D. thesis).

- Shillito, I., van der Giessen, I., Goré, R., Iemhoff, R., 2023. A New Calculus for Intuitionistic Strong Löb Logic: Strong Termination and Cut-Elimination, Formalised, in: Ramanayake, R., Urban, J. (Eds.), Automated Reasoning with Analytic Tableaux and Related Methods. Springer Nature Switzerland, Cham, pp. 73–93.. https://doi.org/10.1007/978-3-031-43513-3_5
- Smoryński, C., 1985. Self-Reference and Modal Logic, Universitext. Springer New York, New York, NY.. <https://doi.org/10.1007/978-1-4613-8601-8>
- Solovay, R.M., 1976. Provability Interpretations of Modal Logic. Israel Journal of Mathematics 25, 287–304.. <https://doi.org/10.1007/BF02757006>
- van der Giessen, I., Iemhoff, R., 2021. Sequent Calculi for Intuitionistic Gödel–Löb Logic. Notre Dame Journal of Formal Logic 62, 221–246.. <https://doi.org/10.1215/00294527-2021-0011>
- Verbrugge, R., 2024. Provability Logic [WWW Document]. The Stanford Encyclopedia of Philosophy (Summer 2024 Edition).. URL <https://plato.stanford.edu/archives/sum2024/entries/logic-provability/> (accessed 7.27.26).
- Visser, A., Litak, T., 2024. Lewis and Brouwer Meet Strong Löb.. <https://doi.org/10.48550/arXiv.2404.11969>
- 舟根大喜, 2024. ZF+ $\neg$ AC の相対的無矛盾性証明の Isabelle/ZF による形式化 [WWW Document].. URL <https://www.math.nagoya-u.ac.jp/~garrigue/tpp2024/slides/funane.pdf>
- 齋藤彰悟, 2024. Lean を用いた Gödel の第一・第二不完全性定理の形式化 [WWW Document].. URL <https://iehalicity.github.io/ffl-slides/main.pdf>