Analysing First-Order Logic in Constructive Type Theory

Dominik Kirst   Dominik Wehr   Yannick Forster

ANU Logic Seminar
December 16th/17th, 2021
What are we working on?

*With members, students and collaborators of the Programming Systems Lab at Saarland University

D. Kirst., D. Wehr, Y. Forster

Analysing FOL in CTT

December 16th/17th, 2021
What are we working on?*

Computability

*With members, students and collaborators of the Programming Systems Lab at Saarland University
What are we working on?*

*With members, students and collaborators of the Programming Systems Lab at Saarland University

Computability

Metamathematics
What are we working on?*

*With members, students and collaborators of the Programming Systems Lab at Saarland University
What are we working on?

Formalisation in Constructive Type Theory

- Computability
- First Order Logic
- Metamathematics

*With members, students and collaborators of the Programming Systems Lab at Saarland University*
What are we working on?*

- Computability
- Metamathematics
- Formalisation in Constructive Type Theory
- Mechanisation in Coq

*With members, students and collaborators of the Programming Systems Lab at Saarland University
What will this talk be about?

First-order logic:

- **Completeness**: classical and intuitionistic first-order logic (Gödel, 1930; Henkin, 1949; Kripke, 1965)
- **Undecidability**: Entscheidungsproblem and Trakhtenbrot’s theorem (Turing, 1938; Trakhtenbrot, 1950)
- **Incompleteness**: Peano arithmetic and ZF set theory (Gödel, 1931; Tarski, 1953)

Constructive type theory:

- **CIC**: Calculus of inductive constructions (Coquand, 1986; Paulin-Mohring, 1993)
- Implementation in the Coq proof assistant (The Coq Development Team, 2021)
- **Synthetic computability** (Richman, 1983; Bauer, 2006)
- **Constructive reverse mathematics** (Ishihara, 2006; Diener, 2020)
What will this talk be about?

First-order logic:
- Completeness: classical and intuitionistic first-order logic (Gödel, 1930; Henkin, 1949; Kripke, 1965)
- Undecidability: Entscheidungsproblem and Trakhtenbrot’s theorem (Turing, 1938; Trakhtenbrot, 1950)
- Incompleteness: Peano arithmetic and ZF set theory (Gödel, 1931; Tarski, 1953)

Constructive type theory:
- CIC: Calculus of inductive constructions (Coquand, 1986; Paulin-Mohring, 1993)
- Implementation in the Coq proof assistant (The Coq Development Team, 2021)
- Synthetic computability (Richman, 1983; Bauer, 2006)
- Constructive reverse mathematics (Ishihara, 2006; Diener, 2020)
Outline

1. First-Order Logic in Coq
2. Completeness
3. Constructive Reverse Mathematics
4. Undecidability: The Entscheidungsproblem
5. More Constructive Reverse Mathematics
6. Undecidability: Trakhtenbrot’s Theorem
7. Relativised Entscheidungsproblem and Incompleteness
8. Conclusion
First-Order Logic in Coq
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

Inductive types:
- 0
- 1
- B
- N
- lists L(X)
- vectors X n...

Simple and dependent type formers:
- X → Y
- X × Y
- X + Y
- ∀ x. F x
- Σ x. F x

Propositional universe P with logical connectives:
- ⊥
- ⊤
- →
- ∧
- ∨
- ∀
- ∃

Specifics of the induced logic:
- Higher-order: quantification over arbitrary functions and relations
- Intuitionistic: does not prove the excluded middle (LEM), stating ∀ P: P ∨ ¬ P.
- Impredicative: ∀ x: X. P x is in P for all P: X → P on arbitrary types X
- Proof-irrelevant: no computational content extractable from ∨ and ∃
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: $0$, $1$, $\mathbb{B}$, $\mathbb{N}$, lists $\mathcal{L}(X)$, vectors $X^n$, ...
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: 0, 1, \( B \), \( \mathbb{N} \), lists \( L(X) \), vectors \( X^n \), ...
- Simple and dependent type formers: \( X \rightarrow Y \), \( X \times Y \), \( X + Y \), \( \forall x. F(x) \), \( \Sigma x. F(x) \)
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: $\emptyset$, $\mathbb{1}$, $\mathbb{B}$, $\mathbb{N}$, lists $L(X)$, vectors $X^n$, ...
- Simple and dependent type formers: $X \rightarrow Y$, $X \times Y$, $X + Y$, $\forall x. F(x)$, $\Sigma x. F(x)$
- Propositional universe $\mathbb{P}$ with logical connectives: $\bot$, $\top$, $\rightarrow$, $\land$, $\lor$, $\forall$, $\exists$
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: 0, 1, ℂ, ℤ, lists \( \mathcal{L}(X) \), vectors \( X^n \), ...
- Simple and dependent type formers: \( X \rightarrow Y \), \( X \times Y \), \( X + Y \), \( \forall x. F x \), \( \Sigma x. F x \)
- Propositional universe \( \mathbb{P} \) with logical connectives: \( \bot \), \( \top \), \( \rightarrow \), \( \land \), \( \lor \), \( \forall \), \( \exists \)

Specifics of the induced logic:
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: 0, 1, B, N, lists L(X), vectors X^n, ...
- Simple and dependent type formers: X → Y, X × Y, X + Y, ∀x. F x, Σx. F x
- Propositional universe P with logical connectives: ⊥, ⊤, →, ∧, ∨, ∀, ∃

Specifics of the induced logic:

- Higher-order: quantification over arbitrary functions and relations
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: $0$, $1$, $\mathbb{B}$, $\mathbb{N}$, lists $\mathcal{L}(X)$, vectors $X^n$, ...
- Simple and dependent type formers: $X \rightarrow Y$, $X \times Y$, $X + Y$, $\forall x. F x$, $\Sigma x. F x$
- Propositional universe $\mathbb{P}$ with logical connectives: $\bot$, $\top$, $\rightarrow$, $\land$, $\lor$, $\forall$, $\exists$

Specifics of the induced logic:

- Higher-order: quantification over arbitrary functions and relations
- Intuitionistic: does not prove the excluded middle (LEM), stating $\forall P : \mathbb{P}. P \lor \neg P$
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: \(0, 1, \mathbb{B}, \mathbb{N},\) lists \(\mathcal{L}(X),\) vectors \(X^n,\) ...
- Simple and dependent type formers: \(X \to Y, X \times Y, X + Y, \forall x. F x, \Sigma x. F x\)
- Propositional universe \(\mathbb{P}\) with logical connectives: \(\bot, \top, \to, \land, \lor, \forall, \exists\)

Specifics of the induced logic:

- Higher-order: quantification over arbitrary functions and relations
- Intuitionistic: does not prove the excluded middle (LEM), stating \(\forall P : \mathbb{P}. P \lor \neg P\)
- Impredicative: \(\forall x : X. P x\) is in \(\mathbb{P}\) for all \(P : X \to \mathbb{P}\) on arbitrary types \(X\)
Calculus of Inductive Constructions (CIC)

Features as implemented in Coq’s type theory:

- Inductive types: 0, 1, B, N, lists L(X), vectors X^n, ...
- Simple and dependent type formers: X → Y, X × Y, X + Y, ∀x. F x, Σx. F x
- Propositional universe P with logical connectives: ⊥, ⊤, →, ∧, ∨, ∀, ∃

Specifics of the induced logic:

- Higher-order: quantification over arbitrary functions and relations
- Intuitionistic: does not prove the excluded middle (LEM), stating ∀P : P. P ∨ ¬P
- Impredicative: ∀x : X. P x is in P for all P : X → P on arbitrary types X
- Proof-irrelevant: no computational content extractable from ∨ and ∃
Representing Syntax

Given a signature $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$, we represent terms and formulas by:

$$t : \text{Term} ::= x \mid ft$$

$$(x : \mathbb{N}, f : \mathcal{F}_\Sigma, \vec{t} : \text{Term}^{|f|})$$

$$\phi, \psi : \text{Form} ::= \bot \mid P\vec{t} \mid \phi \Box \psi \mid \dot{\nabla}\phi$$

$$(P : \mathcal{P}_\Sigma, \vec{t} : \text{Term}^{|P|})$$
Representing Syntax

Given a signature $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$, we represent terms and formulas by:

$$
t : \text{Term} ::= x \mid f \bar{t}$$

$$
\varphi, \psi : \text{Form} ::= \bot \mid P \bar{t} \mid \varphi \Box \psi \mid \nabla \varphi$$

$$
(x : \mathbb{N}, \ f : \mathcal{F}_\Sigma, \ \bar{t} : \text{Term}^{\mathcal{F}_\Sigma})$$

$$
(P : \mathcal{P}_\Sigma, \ \bar{t} : \text{Term}^{\mathcal{P}_\Sigma})$$

De Bruijn representation of binding: $\forall x. \exists y. P(x, y, z) \mapsto \forall \exists P(1, 0, 7)$
Representing Syntax

Given a signature $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$, we represent terms and formulas by:

$$
t : \text{Term} ::= x \mid f \vec{t}$$

$$(x : \mathbb{N}, \ f : \mathcal{F}_\Sigma, \ \vec{t} : \text{Term}^{\mid f \mid})$$

$$\varphi, \psi : \text{Form} ::= \bot \mid P \vec{t} \mid \varphi \Box \psi \mid \Diamond \varphi$$

$$(P : \mathcal{P}_\Sigma, \ \vec{t} : \text{Term}^{\mid P \mid})$$

De Bruijn representation of binding: $\forall x. \exists y. P(x, y, z) \mapsto \forall \exists P(1, 0, 7)$

Coq implementation:

- Separate type classes for signature components $\mathcal{F}_\Sigma$ and $\mathcal{P}_\Sigma$
- Shared type class for operators $\Box$ and $\Diamond$, instances for full syntax and $\forall, \rightarrow$-fragment
- Type class flag for falsity symbol $\bot$
- Tactics for handling de Bruijn substitution
Representing Semantics

A model $\mathcal{M}$ over a domain $D$ is a pair of interpretation functions:

$\mathcal{M} : \forall f : \mathcal{F}_\Sigma. D^{|f|} \rightarrow D$

$\mathcal{M} : \forall P : \mathcal{P}_\Sigma. D^{|P|} \rightarrow \mathbb{P}$
Representing Semantics

A model \( \mathcal{M} \) over a domain \( D \) is a pair of interpretation functions:

\[
\begin{align*}
\mathcal{M} : \forall f : \mathcal{F}_\Sigma. D^{|f|} &\rightarrow D \\
\mathcal{M} : \forall P : \mathcal{P}_\Sigma. D^{|P|} &\rightarrow \mathbb{P}
\end{align*}
\]

For assignments \( \rho : \mathbb{N} \rightarrow D \) define evaluation \( \hat{\rho} t \) and satisfaction \( \mathcal{M} \models \rho \varphi \):

\[
\begin{align*}
\hat{\rho} x &:= \rho x \\
\hat{\rho} (f \bar{t}) &:= f^\mathcal{M} (\hat{\rho} \bar{t}) \\
\mathcal{M} \models \rho P \bar{t} &:= P^\mathcal{M} (\hat{\rho} \bar{t}) \\
\mathcal{M} \models \rho \bot &:= \bot \\
\mathcal{M} \models \rho \varphi \Box \psi &:= \mathcal{M} \models \rho \varphi \Box \mathcal{M} \models \rho \psi \\
\mathcal{M} \models \rho \nabla \varphi &:= \nabla a : D. \mathcal{M} \models a \cdot \rho \varphi
\end{align*}
\]

Obtain derived notation for theories \( \mathcal{T} : \text{Form} \rightarrow \mathbb{P} \) like \( \mathcal{M} \models \mathcal{T} \), and \( \mathcal{T} \models \varphi \).
Representing Semantics

A model \( M \) over a domain \( D \) is a pair of interpretation functions:

\[
\begin{align*}
\mathcal{M} & : \forall f : \mathcal{F}_\Sigma. D^{\lvert f \rvert} \rightarrow D \\
\mathcal{M} & : \forall P : \mathcal{P}_\Sigma. D^{\lvert P \rvert} \rightarrow \mathbb{P}
\end{align*}
\]

For assignments \( \rho : \mathbb{N} \rightarrow D \) define evaluation \( \hat{\rho} t \) and satisfaction \( \mathcal{M} \models_\rho \varphi \):

\[
\begin{align*}
\hat{\rho} x & := \rho x \\
\hat{\rho} (f \overrightarrow{t}) & := f^{\mathcal{M}}(\hat{\rho} \overrightarrow{t}) \\
\mathcal{M} \models_\rho P \overrightarrow{t} & := P^{\mathcal{M}}(\hat{\rho} \overrightarrow{t}) \\
\mathcal{M} \models_\rho \bot & := \bot
\end{align*}
\]

\[
\begin{align*}
\mathcal{M} \models_\rho \varphi \square \psi & := \mathcal{M} \models_\rho \varphi \square \mathcal{M} \models_\rho \psi \\
\mathcal{M} \models_\rho \nabla \varphi & := \nabla a : \mathbb{D}. \mathcal{M} \models_{a,\rho} \varphi
\end{align*}
\]

Obtain derived notation for theories \( T : \text{Form} \rightarrow \mathbb{P} \) like \( \mathcal{M} \models T \), and \( T \models \varphi \).

Coq implementation: type class for models, structurally recursive functions
Representing Deduction

Represented as inductive predicates of the form $\mathcal{L}(\text{Form}) \rightarrow \text{Form} \rightarrow \mathbb{P}$:

\[
\begin{align*}
\Gamma \vdash \forall \phi & \quad \text{AE} \\
\Gamma \vdash \phi[t] & \\
\Gamma \vdash \phi & \quad \text{AI} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash \exists \phi & \\
\Gamma \vdash \exists \phi[t] & \\
\Gamma \vdash \exists \phi & \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash \exists \phi & \\
\Gamma \vdash \exists \phi[t] & \\
\Gamma \vdash \exists \phi & \\
\end{align*}
\]
Representing Deduction

Represented as inductive predicates of the form \( \mathcal{L}(\text{Form}) \rightarrow \text{Form} \rightarrow \mathbb{P} \):

\[
\begin{align*}
\Gamma \vdash \forall \varphi & \quad \frac{\Gamma \vdash \dot{\varphi}}{\Gamma \vdash \varphi[t]} \quad \text{AE} \\
\Gamma \vdash \forall \varphi & \quad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \dot{\forall} \varphi} \quad \text{AI}
\end{align*}
\]

\[ \cdots \]

Given \( \Gamma \) and \( \varphi \) one can compute a fresh variable \( x \) such that \( \Gamma[+1] \vdash \varphi \) iff \( \Gamma \vdash \varphi[x] \)
Representing Deduction

Represented as inductive predicates of the form $\mathcal{L}(\text{Form}) \rightarrow \text{Form} \rightarrow \mathbb{P}$:

\[
\frac{\Gamma \vdash \forall \varphi}{\Gamma \vdash \varphi[t]} \quad \text{AE} \quad \quad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \forall \varphi} \quad \text{AI}
\]

\[
\vdots
\]

- Given $\Gamma$ and $\varphi$ one can compute a fresh variable $x$ such that $\Gamma[+1] \vdash \varphi$ iff $\Gamma \vdash \varphi[x]$
- Switch between intuitionistic variant $\vdash_i$ and classical $\vdash_c$ via $\Gamma \vdash_c ((\varphi \rightarrow \psi) \rightarrow \varphi) \rightarrow \varphi$
Representing Deduction

Represented as inductive predicates of the form $\mathcal{L}(\text{Form}) \rightarrow \text{Form} \rightarrow \mathbb{P}$:

\[
\begin{align*}
\Gamma &\vdash \forall \varphi \\
\Gamma &\vdash \varphi[t] & \text{AE} \\
\Gamma[+1] &\vdash \varphi & \text{AI} \\
\end{align*}
\]

- Given $\Gamma$ and $\varphi$ one can compute a fresh variable $x$ such that $\Gamma[+1] \vdash \varphi$ iff $\Gamma \vdash \varphi[x]$.
- Switch between intuitionistic variant $\vdash_i$ and classical $\vdash_c$ via $\Gamma \vdash_c ((\varphi \rightarrow \psi) \rightarrow \varphi) \rightarrow \varphi$.
- Natural generalisation to $\mathcal{T} \vdash \varphi$ by $\exists \Gamma \subseteq \mathcal{T}. \Gamma \vdash \varphi$. 

D. Kirst., D. Wehr, Y. Forster

Analysing FOL in CTT

December 16th/17th, 2021 9
Representing Deduction

Represented as inductive predicates of the form $L(\text{Form}) \to \text{Form} \to \mathbb{P}$:

\[
\frac{\Gamma \vdash \forall \varphi}{\Gamma \vdash \varphi[t]} \quad AE \quad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \forall \varphi} \quad Al
\]

\[
\ldots
\]

- Given $\Gamma$ and $\varphi$ one can compute a fresh variable $x$ such that $\Gamma[+1] \vdash \varphi$ iff $\Gamma \vdash \varphi[x]$.
- Switch between intuitionistic variant $\vdash_i$ and classical $\vdash_c$ via $\Gamma \vdash_c ((\varphi \to \psi) \to \varphi) \to \varphi$.
- Natural generalisation to $\mathcal{T} \vdash \varphi$ by $\exists \Gamma \subseteq \mathcal{T}. \Gamma \vdash \varphi$.

Coq implementation: type class flag to indicate intuitionistic or classical variant.
Curious Observations

Fact

The intuitionistic deduction system is sound for Tarski semantics, i.e. $\Gamma \vdash i \phi$ implies $\Gamma \models \phi$.

For $\vdash c$ we either need to assume LEM or restrict to classical models validating $\vdash c$.

Constructive reverse mathematics: in fact unrestricted soundness for $\vdash c$ implies LEM.

Fact

For suitable signatures $\Sigma = (F_\Sigma; P_\Sigma)$ one can construct functions $d : Form \rightarrow Form \rightarrow B$ such that $\forall \phi \psi. \phi = \psi \iff d \phi \psi = tt$, $e : N \rightarrow Form$ such that $\forall \phi. \vdash \phi \iff \exists n. e n = \phi$.

Synthetic computability: equality on formulas is decidable ($D$) and provability is enumerable ($E$).
Curious Observations

Fact

*The intuitionistic deduction system is sound for Tarski semantics, i.e. \( \Gamma \vdash_i \phi \) implies \( \Gamma \models \phi \). For \( \vdash_c \) we either need to assume LEM or restrict to classical models validating \( \vdash_c \).*
Curious Observations

Fact

The intuitionistic deduction system is sound for Tarski semantics, i.e. \( \Gamma \vdash_i \varphi \) implies \( \Gamma \vDash \varphi \).
For \( \vdash_c \) we either need to assume LEM or restrict to classical models validating \( \vdash_c \).

Constructive reverse mathematics: in fact unrestricted soundness for \( \vdash_c \) implies LEM.
Curious Observations

Fact

The intuitionistic deduction system is sound for Tarski semantics, i.e. $\Gamma \vdash_i \varphi$ implies $\Gamma \vDash \varphi$. For $\vdash_c$ we either need to assume LEM or restrict to classical models validating $\vdash_c$.

Constructive reverse mathematics: in fact unrestricted soundness for $\vdash_c$ implies LEM

Fact

For suitable signatures $\Sigma = (F_\Sigma; P_\Sigma)$ one can construct functions

- $d : \text{Form} \to \text{Form} \to \mathbb{B}$ such that $\forall \varphi \psi. \varphi = \psi \iff d \varphi \psi = \text{tt}$,
- $e : \mathbb{N} \to \text{Form}$ such that $\forall \varphi. \vdash \varphi \iff \exists n. e n = \varphi$. 

Curious Observations

Fact

*The intuitionistic deduction system is sound for Tarski semantics, i.e. \( \Gamma \vdash_i \varphi \) implies \( \Gamma \models \varphi \). For \( \vdash_c \) we either need to assume LEM or restrict to classical models validating \( \vdash_c \).*

Constructive reverse mathematics: in fact unrestricted soundness for \( \vdash_c \) implies LEM

Fact

*For suitable signatures \( \Sigma = (F_\Sigma; P_\Sigma) \) one can construct functions*

- \( d : \text{Form} \to \text{Form} \to \mathbb{B} \) such that \( \forall \varphi \psi. \varphi = \psi \leftrightarrow d \varphi \psi = \text{tt} \),
- \( e : \mathbb{N} \to \text{Form} \) such that \( \forall \varphi. \vdash \varphi \leftrightarrow \exists n. e n = \varphi \).

Synthetic computability: equality on formulas is decidable (\( D \)) and provability is enumerable (\( E \))
Completeness*
Completeness: Syntax fragments

A lot of our work is restricted to the fragment

\[
\begin{align*}
t : \text{Term} & ::= x \mid f \bar{t} & n : \mathbb{N}, f : \Sigma \\
\varphi : \text{Form}^* & ::= \top \mid P \bar{t} \mid \varphi \rightarrow \psi \mid \forall \varphi & P : \Sigma
\end{align*}
\]
Completeness: Syntax fragments

A lot of our work is restricted to the fragment

\[
\begin{align*}
  t : \text{Term} &::= x \mid f \bar{t} & n : \mathbb{N}, f : \Sigma \\
  \varphi : \text{Form}^* &::= \bot \mid P \bar{t} \mid \varphi \rightarrow \psi \mid \forall \varphi & P : \Sigma
\end{align*}
\]

Definition

A model \( \mathcal{M} \) consists of a type \( X \)

- ... 
- an absurdity interpretation \( \bot^\mathcal{M} : \mathbb{P} \).

Interpreting

\[
\mathcal{M} \models_\rho \bot :\iff \bot^\mathcal{M}
\]
Tarski models: Classes of models

Definition

A model $\mathcal{M}$ is called classical* if for all $\rho$ and $\varphi : \text{Form}^*$

$$\mathcal{M} \models \rho \neg \neg \varphi \rightarrow \varphi$$

A model $\mathcal{M}$ is called standard if $\bot$ is contradictory.

A model $\mathcal{M}$ is called exploding* (due to Veldman (1976)) if for all $\rho$ and formulas $\varphi$:

$$\mathcal{M} \models \rho \neg \neg \varphi \rightarrow \varphi$$

Fact

Every standard model is exploding*. The converse need not hold.
Tarski models: Classes of models

Definition

A model $\mathcal{M}$ is called **classical** if for all $\rho$ and $\varphi$:

$$\mathcal{M} \models_\rho \lnot \lnot \varphi \rightarrow \varphi$$

A model $\mathcal{M}$ is called **standard** if $\bot^\mathcal{M}$ is contradictory.
## Definition

A model $\mathcal{M}$ is called **classical** if for all $\rho$ and $\varphi : \text{Form}$:

$$\mathcal{M} \models \rho \not\vdash \varphi \rightarrow \varphi$$

A model $\mathcal{M}$ is called **standard** if $\bot_\mathcal{M}$ is contradictory.

A model $\mathcal{M}$ is called **exploding** (due to Veldman (1976)) if for all $\rho$ and formulas $\varphi : \text{Form}$:

$$\mathcal{M} \models \rho \not\vdash \bot_\mathcal{M} \rightarrow \varphi$$
Tarski models: Classes of models

Definition

A model $\mathcal{M}$ is called **classical** if for all $\rho$ and $\varphi : \text{Form}^*$

$$\mathcal{M} \models_{\rho} \bot \rightarrow \varphi$$

A model $\mathcal{M}$ is called **standard** if $\bot^\mathcal{M}$ is contradictory.

A model $\mathcal{M}$ is called **exploding** (due to Veldman (1976)) if for all $\rho$ and formulas $\varphi : \text{Form}^*$

$$\mathcal{M} \models_{\rho} \bot \rightarrow \varphi$$

Fact

*Every standard model is exploding*. The converse need not hold.
Tarski completeness: Model construction

Lemma

Pick a closed theory $T$. There exists a model $M, \rho$ such that:

- If $T \vdash \varphi$ then $M \models_\rho \varphi$
- If $M \models_\rho \bot$ then $T \vdash \bot$

Proof.

Extend $T$ into a theory $\Omega$ as follows:

1. Henkin axioms: Add all formulas $\varphi_n(n \rightarrow \forall \varphi_n)$
2. Lindenbaum: Add all formulas maintaining consistency

The term-model induced by $\Omega$ fulfills all desiderata.
Tarski completeness: Model construction

Lemma

*Pick a closed theory $\mathcal{T}$. There exists a model $\mathcal{M}, \rho$ such that:*

- If $\mathcal{T} \vdash \varphi$ then $\mathcal{M} \models_\rho \varphi$
- If $\mathcal{M} \models_\rho \bot$ then $\mathcal{T} \vdash \bot$

Proof.

Extend $\mathcal{T}$ into a theory $\Omega$ as follows:
Tarski completeness: Model construction

Lemma

Pick a closed theory $\mathcal{T}$. There exists a model $\mathcal{M}, \rho$ such that:

- If $\mathcal{T} \vdash \varphi$ then $\mathcal{M} \models_{\rho} \varphi$
- If $\mathcal{M} \models_{\rho} \bot$ then $\mathcal{T} \vdash \bot$

Proof.

Extend $\mathcal{T}$ into a theory $\Omega$ as follows:

1. Henkin axioms: Add all formulas $\varphi_n(n) \rightarrow \forall \varphi_n$
Tarski completeness: Model construction

Lemma

*Pick a closed theory $\mathcal{T}$. There exists a model $\mathcal{M}, \rho$ such that:*

- If $\mathcal{T} \vdash \varphi$ then $\mathcal{M} \models \rho \varphi$ 
- If $\mathcal{M} \models \rho \bot$ then $\mathcal{T} \vdash \bot$

Proof.

Extend $\mathcal{T}$ into a theory $\Omega$ as follows:

1. Henkin axioms: Add all formulas $\varphi_n(n) \rightarrow \forall \varphi_n$
2. Lindenbaum: Add all formulas maintaining consistency
Tarski completeness: Model construction

**Lemma**

*Pick a closed theory \( T \). There exists a model \( M, \rho \) such that:*

- If \( T \vdash \varphi \) then \( M \models \rho \varphi \)
- If \( M \models \rho \bot \) then \( T \vdash \bot \)

**Proof.**

Extend \( T \) into a theory \( \Omega \) as follows:

1. Henkin axioms: Add all formulas \( \varphi_n(n) \to \forall \varphi_n \)
2. Lindenbaum: Add all formulas maintaining consistency

The term-model induced by \( \Omega \) fulfills all desiderata.
Tarski completeness: The standard case

Theorem

For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \models \varphi$ entails $\neg\neg(\mathcal{T} \vdash \varphi)$.
Theorem

For closed $T$, $\varphi$ we know $T \models \varphi$ entails $\neg \neg (T \vdash \varphi)$.

Proof.
Tarski completeness: The standard case

**Theorem**

*For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \models \varphi$ entails $\neg\neg(\mathcal{T} \vdash \varphi)$.*

**Proof.**

Suppose $\mathcal{T} \not\vdash \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\neg \varphi\}$ is consistent.
Tarski completeness: The standard case

**Theorem**

*For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \vDash \varphi$ entails $\neg \neg (\mathcal{T} \vdash \varphi)$.*

**Proof.**

Suppose $\mathcal{T} \not\vDash \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\neg \varphi\}$ is consistent. Applying the lemma to $\mathcal{T}'$ yields

1. If $\mathcal{T} \cup \{\neg \varphi\} \vDash \varphi$ then $\mathcal{M} \vDash_{\rho} \varphi$
2. If $\mathcal{M} \vDash_{\rho} \bot$ then $\mathcal{T} \cup \{\neg \varphi\} \vDash \bot$

---

D. Kirst., D. Wehr, Y. Forster

**Analysing FOL in CTT**

December 16th/17th, 2021 15
Tarski completeness: The standard case

**Theorem**

For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \models \varphi$ entails $\neg\neg(\mathcal{T} \vdash \varphi)$.

**Proof.**

Suppose $\mathcal{T} \not\vdash \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\neg\varphi\}$ is consistent. Applying the lemma to $\mathcal{T'}$ yields

1. If $\mathcal{T} \cup \{\neg\varphi\} \vdash \varphi$ then $\mathcal{M} \models_{\rho} \varphi$
2. If $\mathcal{M} \models_{\rho} \bot$ then $\mathcal{T} \cup \{\neg\varphi\} \vdash \bot$

Then

- By (1) $\mathcal{M}, \rho$ is classical* and by (2) $\mathcal{M}, \rho$ is standard.
Tarski completeness: The standard case

**Theorem**

For closed \( T \), \( \varphi \) we know \( T \vDash \varphi \) entails \( \neg \neg (T \vdash \varphi) \).

**Proof.**

Suppose \( T \not\vdash \varphi \), meaning \( T' := T \cup \{\neg \varphi\} \) is consistent. Applying the lemma to \( T' \) yields

1. If \( T \cup \{\neg \varphi\} \vdash \varphi \) then \( M \vDash \varphi \)
2. If \( M \vDash \varphi \) then \( T \cup \{\neg \varphi\} \vdash \bot \)

Then

- By (1) \( M, \rho \) is classical* and by (2) \( M, \rho \) is standard.
- By (1) we have \( M \vDash \varphi \) and thus \( M \vDash \varphi \).

\[ \square \]
Tarski completeness: The standard case

**Theorem**

For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \models \varphi$ entails $\neg\neg(\mathcal{T} \vdash \varphi)$.

**Proof.**

Suppose $\mathcal{T} \nvdash \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\neg \varphi\}$ is consistent. Applying the lemma to $\mathcal{T}'$ yields

1. If $\mathcal{T} \cup \{\neg \varphi\} \vdash \varphi$ then $\mathcal{M} \models_\rho \varphi$
2. If $\mathcal{M} \models_\rho \bot$ then $\mathcal{T} \cup \{\neg \varphi\} \nvdash \bot$

Then

- By (1) $\mathcal{M}, \rho$ is classical* and by (2) $\mathcal{M}, \rho$ is standard.
- By (1) we have $\mathcal{M} \models_\rho \mathcal{T}$ and thus $\mathcal{M} \models_\rho \varphi$.
- But by (1) we also have $\mathcal{M} \models_\rho \neg \varphi$. 

\[ \square \]
Tarski completeness: The standard case

**Theorem**

*For closed $T$, $\varphi$ we know $T \models \varphi$ entails $\neg\neg(T \vdash \varphi).$*

**Fact**

$$T \models \varphi \rightarrow T \vdash \varphi \quad \text{iff} \quad \neg\neg T \vdash \varphi \rightarrow T \vdash \varphi.$$
Tarski completeness: The standard case

**Theorem**

*For closed $T$, $\varphi$ we know $T \models \varphi$ entails $\neg\neg(T \vdash \varphi)$.*

**Fact**

$T \models \varphi \rightarrow T \vdash \varphi$  iff  $\neg\neg T \vdash \varphi \rightarrow T \vdash \varphi$.

**Corollary**

$\forall T, \varphi. \ T \models \varphi \rightarrow T \vdash \varphi$ is not provable in the CIC.
Tarski completeness: Examining the standard case

Theorem

For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \vDash \varphi$ entails $\neg\neg(\mathcal{T} \vdash \varphi)$.

Proof.

Suppose $\mathcal{T} \not\vDash \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\neg \varphi\}$ is consistent. Applying the lemma to $\mathcal{T}'$ yields

1. If $\mathcal{T}' \cup \{\neg \varphi\} \vdash \varphi$ then $\mathcal{M} \vDash_{\varphi} \varphi$
2. If $\mathcal{M} \vDash_{\varphi} \bot$ then $\mathcal{T}' \cup \{\neg \varphi\} \vdash \bot$

Then

- By (1) $\mathcal{M}, \rho$ is classical* and by (2) $\mathcal{M}, \rho$ is standard.
- By (1) we have $\mathcal{M} \vDash_{\varphi} \mathcal{T}$ and thus $\mathcal{M} \vDash_{\varphi} \varphi$.
- But by (1) we also have $\mathcal{M} \vDash_{\varphi} \neg \varphi$.

*Following Herbelin and Ilik (2016)
Tarski completeness: Examining the standard case*

Theorem

For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \models \varphi$ entails $\neg\neg(\mathcal{T} \vdash \varphi)$.

Proof.

Suppose $\mathcal{T} \not\models \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\neg \varphi\}$ is consistent. Applying the lemma to $\mathcal{T}'$ yields

1. If $\mathcal{T} \cup \{\neg \varphi\} \vdash \varphi$ then $\mathcal{M} \vDash \rho \varphi$
2. If $\mathcal{M} \vDash \rho \bot$ then $\mathcal{T} \cup \{\neg \varphi\} \vdash \bot$

Then

- By (1) $\mathcal{M}, \rho$ is classical* and by (2) $\mathcal{M}, \rho$ is standard.
- By (1) we have $\mathcal{M} \vDash \rho \mathcal{T}$ and thus $\mathcal{M} \vDash \rho \varphi$.
- But by (1) we also have $\mathcal{M} \vDash \rho \neg \varphi$.

*Following Herbelin and Ilik (2016)
Tarski completeness: Examining the standard case

Theorem

For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \vDash \varphi$ entails $\neg\neg(\mathcal{T} \vdash \varphi)$.

Proof.

Suppose $\mathcal{T} \not\vDash \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\neg \varphi\}$ is consistent. Applying the lemma to $\mathcal{T}'$ yields

1. If $\mathcal{T} \cup \{\neg \varphi\} \vdash \varphi$ then $\mathcal{M} \vDash _{\rho} \varphi$
2. If $\mathcal{M} \vDash _{\rho} \bot$ then $\mathcal{T} \cup \{\neg \varphi\} \vdash \bot$

Then

- By (1) $\mathcal{M}, \rho$ is classical* and by (2) $\mathcal{M}, \rho$ is standard.
- By (1) we have $\mathcal{M} \vDash _{\rho} \mathcal{T}$ and thus $\mathcal{M} \vDash _{\rho} \varphi$.
- But by (1) we also have $\mathcal{M} \vDash _{\rho} \neg \varphi$.

*Following Herbelin and Ilik (2016)
Theorem

For closed $\mathcal{T}$, $\varphi$ we know $\mathcal{T} \models^E \varphi$ entails $\mathcal{T} \vdash \varphi$.

Proof.

Applying the lemma to $\mathcal{T}' := \mathcal{T} \cup \{\neg \varphi\}$ yields

1. If $\mathcal{T} \cup \{\neg \varphi\} \vdash \varphi$ then $\mathcal{M} \models_\rho \varphi$
2. If $\mathcal{M} \models_\rho \bot$ then $\mathcal{T} \cup \{\neg \varphi\} \vdash \bot$

Then

- By (1) $\mathcal{M}, \rho$ is classical* and $\mathcal{M}, \rho$ exploding*.
- By (1) we have $\mathcal{M} \models_\rho \mathcal{T}$ and thus $\mathcal{M} \models_\rho \varphi$.
- By (1) we also have $\mathcal{M} \models_\rho \neg \varphi$, meaning $\mathcal{M} \models_\rho \bot$.
- By (2) this means $\mathcal{T} \cup \{\neg \varphi\} \vdash \bot$ and thus $\mathcal{T} \vdash \varphi$.
Kripke structures

**Definition**

A Kripke structure $\mathcal{K}$ consists of a preorder $(\mathcal{W}, \leq)$, a domain $D : \mathbb{T}$ and

- for each $f : \Sigma$ an interpretation $f : D^{\mid f \mid} \rightarrow D$
- for each $P : \Sigma$ and $w : \mathcal{W}$ an interpretation $P_w : D^{\mid P \mid} \rightarrow \mathbb{P}$
- for each world $w : \mathcal{W}$ an interpretation $\bot_w : \mathbb{P}$

such that, if $w \leq v$ then

- $\bot_w$ entails $\bot_v$
- $P_w \vec{d}$ entails $P_v \vec{d}$ for all $\vec{d} : D^{\mid P \mid}$

Note: Such models are only complete on the Form $\star$-fragment!
## Kripke structures

### Definition

A Kripke structure $\mathcal{K}$ consists of a preorder $(\mathcal{W}, \leq)$, a domain $D : \mathbb{T}$ and

- for each $f : \Sigma$ an interpretation $f : D^{|f|} \to D$
- for each $P : \Sigma$ and $w : \mathcal{W}$ an interpretation $P_w : D^{|P|} \to \mathbb{P}$
- for each world $w : \mathcal{W}$ an interpretation $\bot_w : \mathbb{P}$

such that, if $w \leq v$ then

- $\bot_w$ entails $\bot_v$
- $P_w \vec{d}$ entails $P_v \vec{d}$ for all $\vec{d} : D^{|P|}$

**Note:** Such models are only complete on the Form*-fragment!
Kripke models: Semantics

Definition

Fix a Kripke structure $\mathcal{K}$. For an assignment $\rho : \mathcal{V} \rightarrow D$ we define

\[
\begin{align*}
    w \models_{\rho} \bot & :\iff \bot_w \\
    w \models_{\rho} P \vec{t} & :\iff P_w \vec{t}^\rho \\
    w \models_{\rho} \forall \varphi & :\iff \forall d : D. \ w \models_{d,\rho} \varphi \\
    w \models_{\rho} \varphi \rightarrow \psi & :\iff \forall w \leq v. \ v \models_{\rho} \varphi \rightarrow v \models_{\rho} \psi
\end{align*}
\]
Kripke models: Semantics

Definition

Fix a Kripke structure $\mathcal{K}$. For an assignment $\rho : \mathcal{V} \to D$ we define

\[
\begin{align*}
    w \models_\rho \forall \varphi & \iff \forall d : D. \ w \models_{d, \rho} \varphi \\
    w \models_\rho \varphi \to \psi & \iff \forall w \leq v. \ v \models_\rho \varphi \to v \models_\rho \psi
\end{align*}
\]
Kripke models: Semantics

Definition

Fix a Kripke structure $\mathcal{K}$. For an assignment $\rho : \mathcal{V} \to D$ we define

- $w \models_\rho \bot : \iff \bot_w$
- $w \models_\rho P \vec{t} : \iff P_w \vec{t}^\rho$
- $w \models_\rho \forall \varphi : \iff \forall d : D. \ w \models_{d, \rho} \varphi$
- $w \models_\rho \varphi \rightarrow \psi : \iff \forall w \leq v. \ v \models_\rho \varphi \rightarrow v \models_\rho \psi$

Fact

If $w \leq v$ then $w \models_\rho \varphi$ entails $v \models_\rho \varphi$. 
Kripke models: Classes of models

**Definition**

A Kripke model $\mathcal{K}, \rho$ is called **exploding** if

$$w \vDash_\rho \bot \rightarrow \varphi$$

for every $w : \mathcal{W}$ and $\varphi$. 

A $\mathcal{K}, \rho$ is called **standard** if $\bot \rightarrow \bot$ for every $w : \mathcal{W}$. 

**Definition**

We write $\Gamma \vDash e \varphi$ if for all exploding $\mathcal{K}, \rho$ and $w : \mathcal{W}$

$$(\forall \psi \in \Gamma. \ w \vDash_\rho \psi) \rightarrow \ w \vDash_\rho \varphi$$

We write $\Gamma \vDash s \varphi$ if the analogous case holds for all standard $\mathcal{K}, \rho$. 

Kripke models: Classes of models

**Definition**

A Kripke model $\mathcal{K}, \rho$ is called **exploding** if

$$w \vDash \rho \ \bot \rightarrow \varphi \text{ for every } w : \mathcal{W} \text{ and } \varphi.$$ 

A $\mathcal{K}, \rho$ is called **standard** if $\bot_w \rightarrow \bot$ for every $w : \mathcal{W}$. 

D. Kirst., D. Wehr, Y. Forster

Analysing FOL in CTT

December 16th/17th, 2021 20
Kripke models: Classes of models

Definition

A Kripke model $K, \rho$ is called **exploding** if

$$w \models_{\rho} \bot \rightarrow \varphi \text{ for every } w : \mathcal{W} \text{ and } \varphi.$$ 

A $K, \rho$ is called **standard** if $\bot_w \rightarrow \bot$ for every $w : \mathcal{W}$.

Definition

We write $\Gamma \models^e \varphi$ if for all exploding $K, \rho$ and $w : \mathcal{W}$

$$\left( \forall \psi \in \Gamma. w \models_{\rho} \psi \right) \rightarrow w \models_{\rho} \varphi$$

We write $\Gamma \models^s \varphi$ if the analogous case holds for all standard $K, \rho$. 

LJT: Focused Sequents

\begin{align*}
\rightarrow_L &: \Gamma \Rightarrow \psi \quad \Gamma; \theta \Rightarrow \phi \\
&\quad \frac{}{\Gamma; \psi \Rightarrow \theta \Rightarrow \phi}
\end{align*}

\begin{align*}
\forall_L &: \Gamma; \psi[t] \Rightarrow \phi \\
&\quad \frac{}{\Gamma; \forall \psi \Rightarrow \phi}
\end{align*}

\begin{align*}
\rightarrow_R &: \Gamma, \phi \Rightarrow \psi \\
&\quad \frac{}{\Gamma \Rightarrow \phi \Rightarrow \psi}
\end{align*}

\begin{align*}
\forall_R &: \Gamma \Rightarrow \forall \phi \\
&\quad \frac{}{\Gamma \Rightarrow \phi}
\end{align*}

\begin{align*}
\text{Ax} &: \Gamma; \varphi \Rightarrow \varphi
\end{align*}

\begin{align*}
\text{Focus} &: \Gamma; \psi \Rightarrow \phi \quad \psi \in \Gamma \\
&\quad \frac{}{\Gamma \Rightarrow \phi}
\end{align*}

\begin{align*}
\text{Exp} &: \Gamma \Rightarrow \perp \\
&\quad \frac{}{\Gamma \Rightarrow \varphi}
\end{align*}
Kripke completeness: The exploding case*

**Definition**

We define the universal structure \( U \) on the preorder \((\mathcal{L}(\text{Form}), \subseteq)\) and the domain Term, taking

\[
P_{\Gamma} t := \Gamma \Rightarrow P \vec{t} \quad \bot_{\Gamma} := \Gamma \Rightarrow \bot
\]

---

*Following Herbelin and Lee (2009)
Kripke completeness: The exploding case*

**Definition**

We define the universal structure \( \mathcal{U} \) on the preorder \((\mathcal{L}(\text{Form}), \subseteq)\) and the domain Term, taking

\[
P_{\Gamma} \vec{t} := \Gamma \Rightarrow P \vec{t} \quad \bot_{\Gamma} := \Gamma \Rightarrow \bot
\]

**Fact**

\( \mathcal{U}, \sigma \) is exploding but not standard.

*Following Herbelin and Lee (2009)*
Kripke completeness: The exploding case*

**Definition**

We define the universal structure $\mathcal{U}$ on the preorder $(\mathcal{L}(\text{Form}), \subseteq)$ and the domain Term, taking

\[ P_\Gamma t := \Gamma \Rightarrow P t \quad \bot_\Gamma := \Gamma \Rightarrow \bot \]

**Fact**

$\mathcal{U}, \sigma$ is exploding but not standard.

**Lemma**

Over the structure $\mathcal{U}$ we have

1. $\Gamma \vdash_\sigma \varphi \rightarrow \Gamma \Rightarrow \varphi[\sigma]$
2. $(\forall \psi, \Gamma \subseteq \Delta$. $\Delta; \varphi[\sigma] \Rightarrow \psi \rightarrow \Delta \Rightarrow \psi) \rightarrow \Gamma \vdash_\sigma \varphi$

*Following Herbelin and Lee (2009)*
Lemma

Over the structure $\mathcal{U}$ we have

1. $\Gamma \vdash_\sigma \varphi \rightarrow \Gamma \Rightarrow \varphi[\sigma]$

2. $(\forall \psi, \Gamma \subseteq \Delta. \varphi[\sigma] \Rightarrow \psi \rightarrow \Delta \Rightarrow \psi) \rightarrow \Gamma \vdash_\sigma \varphi$

Corollary

If $\Gamma \vdash e \varphi$ then $\Gamma \Rightarrow \varphi$.

Proof.

Work within the model $\mathcal{U}$: We know $\Delta \vdash \sigma \Gamma$ entails $\Delta \vdash \sigma \varphi$ for any $\Delta$.

Using (2) we conclude $\Gamma \vdash id \Gamma$. Per assumption thus $\Gamma \vdash id \varphi$ and $\Gamma \Rightarrow \varphi$ by (1).
Kripke completeness: The exploding case II

Lemma

Over the structure $\mathcal{U}$ we have

1. $\Gamma \vdash_\sigma \varphi \rightarrow \Gamma \Rightarrow \varphi[\sigma]$ 
2. $(\forall \psi, \Gamma \subseteq \Delta. \Delta; \varphi[\sigma] \Rightarrow \psi \rightarrow \Delta \Rightarrow \psi) \rightarrow \Gamma \vdash_\sigma \varphi$

Corollary

If $\Gamma \vdash^e \varphi$ then $\Gamma \Rightarrow \varphi$. 

Proof. Work within the model $\mathcal{U}$: We know $\Delta \vdash \sigma \Gamma$ entails $\Delta \vdash \sigma \varphi$ for any $\Delta$.

Using (2) we conclude $\Gamma \vdash \text{id}_\Gamma$. Per assumption thus $\Gamma \vdash \text{id}_\varphi$ and $\Gamma \Rightarrow \varphi$ by (1).
Lemma

Over the structure $\mathcal{U}$ we have

1. $\Gamma \models_\sigma \varphi \rightarrow \Gamma \Rightarrow \varphi[\sigma]$
2. $(\forall \psi, \Gamma \subseteq \Delta. \Delta; \varphi[\sigma] \Rightarrow \psi \rightarrow \Delta \Rightarrow \psi) \rightarrow \Gamma \models_\sigma \varphi$

Corollary

If $\Gamma \models^e \varphi$ then $\Gamma \Rightarrow \varphi$.

Proof.

Work within the model $\mathcal{U}$: We know $\Delta \models_\sigma \Gamma$ entails $\Delta \models_\sigma \varphi$ for any $\Delta$. 

D. Kirst., D. Wehr, Y. Forster  
Analysing FOL in CTT  
December 16th/17th, 2021 23
Lemma

Over the structure $\mathcal{U}$ we have

1. $\Gamma \models_\sigma \varphi \rightarrow \Gamma \Rightarrow \varphi[\sigma]$
2. $(\forall \psi, \Gamma \subseteq \Delta. \Delta; \varphi[\sigma] \Rightarrow \psi \rightarrow \Delta \Rightarrow \psi) \rightarrow \Gamma \models_\sigma \varphi$

Corollary

If $\Gamma \models^e \varphi$ then $\Gamma \Rightarrow \varphi$.

Proof.

Work within the model $\mathcal{U}$: We know $\Delta \models_\sigma \Gamma$ entails $\Delta \models_\sigma \varphi$ for any $\Delta$.

- Using (2) we conclude $\Gamma \models_{\text{id}} \Gamma$
Kripke completeness: The exploding case II

Lemma

*Over the structure* \( \mathcal{U} \) *we have*

1. \( \Gamma \vdash_\sigma \varphi \rightarrow \Gamma \Rightarrow \varphi[\sigma] \)
2. \( (\forall \psi, \Gamma \subseteq \Delta. \Delta; \varphi[\sigma] \Rightarrow \psi \rightarrow \Delta \Rightarrow \psi) \rightarrow \Gamma \vdash_\sigma \varphi \)

Corollary

*If* \( \Gamma \vdash^e \varphi \) *then* \( \Gamma \Rightarrow \varphi. \)

Proof.

Work within the model \( \mathcal{U} \): We know \( \Delta \vdash_\sigma \Gamma \) entails \( \Delta \vdash_\sigma \varphi \) for any \( \Delta \).

- Using (2) we conclude \( \Gamma \vdash_{\text{id}} \Gamma \)
- Per assumption thus \( \Gamma \vdash_{\text{id}} \varphi \) and \( \Gamma \Rightarrow \varphi \) by (1)
Kripke completeness: The standard case

**Definition**

We define the consistent structure $C$ on the preorder $(\Sigma \Gamma : \mathcal{L}(\text{Form})$. $\Gamma \nvdash \bot$, $\subseteq$) and the domain $\text{Term}$, taking

$$P_{\Gamma \vec{t}} := \neg \neg (\Gamma \Rightarrow P \vec{t}) \quad \bot_{\Gamma} := \Gamma \Rightarrow \bot$$
Kripke completeness: The standard case

**Definition**

We define the consistent structure $C$ on the preorder $(\Sigma \Gamma : L(\text{Form}). \Gamma \not\models \bot, \subseteq)$ and the domain Term, taking

\[ P_{\Gamma} \vec{t} := \neg \neg (\Gamma \Rightarrow P \vec{t}) \quad \bot_{\Gamma} := \Gamma \Rightarrow \bot \]

**Fact**

$U, \sigma$ is a standard model.
Kripke completeness: The standard case

Definition

We define the consistent structure $C$ on the preorder $(\Sigma \Gamma : L(\text{Form}). \Gamma \not\vdash \bot, \subseteq)$ and the domain Term, taking

$$P_{\Gamma \vec{t}} := \neg \neg (\Gamma \Rightarrow P \vec{t}) \quad \bot_{\Gamma} := \Gamma \Rightarrow \bot$$

Fact

$\mathcal{U}, \sigma$ is a standard model.

Fact

$\Gamma \vDash^- \varphi \rightarrow \Gamma \Rightarrow \varphi$ iff $\neg \neg (\Gamma \vdash \varphi) \rightarrow \Gamma \vdash \varphi$. 

D. Kirst., D. Wehr, Y. Forster
Completeness: Other semantics

The following semantics admit constructive completeness proofs wrt. the full syntax:

- Heyting algebras wrt. intuitionistic FOL
- Intuitionistic formal dialogues wrt. intuitionistic FOL
- Classical formal dialogues wrt. classical FOL
- Classical material dialogues wrt. classical FOL

Furthermore

- Assuming the EM, we can obtain completeness for classical, standard Tarski models
- Intuitionistic material dialogues are incomplete over CIC
Constructive Reverse Mathematics *

*F., K., and W. at LFCS’20.
Constructive Reverse Mathematics *

or: Find axioms equivalent to theorems.
Sufficient and necessary axioms for completeness

Is there a well-known axiom $A$ with

$$A \rightarrow \forall \Gamma \varphi. \Gamma \models \varphi \rightarrow \Gamma \vdash \varphi$$
Sufficient and necessary axioms for completeness

Is there a well-known axiom $A$ with

$$A \iff \forall \Gamma \varphi. \Gamma \models \varphi \rightarrow \Gamma \vdash \varphi$$
Sufficient and necessary axioms for completeness

Is there a well-known axiom $A$ with

\[ A \leftrightarrow \forall \Gamma \varphi. \neg (\Gamma \not\vdash \varphi) \rightarrow \Gamma \vdash \varphi \]
Sufficient and necessary axioms for completeness

Are there well-known axioms $A$ and $A_2$ with

$$\begin{align*}
A & \iff \forall \Gamma. \neg(\Gamma \not\vdash \varphi) \rightarrow \Gamma \vdash \varphi \\
A_2 & \iff \forall T. \neg(T \not\vdash \varphi) \rightarrow T \vdash \varphi
\end{align*}$$
Is there a well-known axiom-scheme $A$ with

$$A(P) \iff \forall T. PT \rightarrow \neg (T \vdash \varphi) \rightarrow T \vdash \varphi$$
Kreisel (1962): Use Markov’s principle MP (proof idea due to Gödel)
Slightly confusing literature review

Kreisel (1962): Use Markov’s principle MP (proof idea due to Gödel)
How to define MP?

\[ MP \; := \; \forall f : \mathbb{N} \rightarrow \mathbb{B}. f \text{ is primitive recursive} \rightarrow \neg \neg(\exists n. fn = \text{true}) \rightarrow \exists n. fn = \text{true} \]
Kreisel (1962): Use Markov’s principle MP (proof idea due to Gödel)

How to define MP?

\[
\text{MP} := \forall f : \mathbb{N} \to \mathbb{B}. f \text{ is primitive recursive } \rightarrow \neg \neg (\exists n. fn = \text{true}) \rightarrow \exists n. fn = \text{true}
\]

\[
\text{MP} := \forall f : \mathbb{N} \to \mathbb{B}. f \text{ is recursive } \rightarrow \neg \neg (\exists n. fn = \text{true}) \rightarrow \exists n. fn = \text{true}
\]
Kreisel (1962): Use Markov’s principle MP (proof idea due to Gödel)
How to define MP?

\[
\text{MP} := \forall f : \mathbb{N} \to \mathbb{B}. f \text{ is primitive recursive } \to \neg\neg(\exists n. fn = \text{true}) \to \exists n. fn = \text{true}
\]

\[
\text{MP} := \forall f : \mathbb{N} \to \mathbb{B}. f \text{ is recursive } \to \neg\neg(\exists n. fn = \text{true}) \to \exists n. fn = \text{true}
\]

\[
\text{MP} := \forall f : \mathbb{N} \to \mathbb{B}. \neg\neg(\exists n. fn = \text{true}) \to \exists n. fn = \text{true}
\]
Slightly confusing literature review

Kreisel (1962): Use Markov’s principle MP (proof idea due to Gödel)

How to define MP?

\[
\text{MP} : = \forall f : \mathbb{N} \rightarrow \mathbb{B}. f \text{ is primitive recursive} \rightarrow \neg\neg(\exists n. fn = \text{true}) \rightarrow \exists n. fn = \text{true}
\]

\[
\text{MP} : = \forall f : \mathbb{N} \rightarrow \mathbb{B}. f \text{ is recursive} \rightarrow \neg\neg(\exists n. fn = \text{true}) \rightarrow \exists n. fn = \text{true}
\]

\[
\text{MP} : = \forall f : \mathbb{N} \rightarrow \mathbb{B}. \neg\neg(\exists n. fn = \text{true}) \rightarrow \exists n. fn = \text{true}
\]

\[
\text{MP} : = \forall X. X \text{ is discrete} \rightarrow \forall p : X \rightarrow \mathcal{P}. \exists p \rightarrow \forall x. \neg\neg px \rightarrow px
\]
Kreisel (1962): Use Markov’s principle MP (proof idea due to Gödel)
How to define MP?

\[ \text{MP}_{pr} := \forall f : \mathbb{N} \to \mathbb{B}. f \text{ is primitive recursive} \to \neg \neg (\exists n. f_n = \text{true}) \to \exists n. f_n = \text{true} \]

\[ \text{MP}_L := \forall f : \mathbb{N} \to \mathbb{B}. f \text{ is recursive} \to \neg \neg (\exists n. f_n = \text{true}) \to \exists n. f_n = \text{true} \]

\[ \text{MP}_{TT} := \forall f : \mathbb{N} \to \mathbb{B}. \neg \neg (\exists n. f_n = \text{true}) \to \exists n. f_n = \text{true} \]

\[ \text{MP}_E := \forall X. X \text{ is discrete} \to \forall p : X \to \mathbb{P}. E_p \to \forall x. \neg \neg p x \to px \]
LEM

\[ \downarrow \]

\[ M_{TT} \]

\[ \leftarrow \]

\[ M_{\varepsilon} \]

\[ \rightarrow \]

\[ M_{pr} \]

\[ \rightarrow \]

\[ M_{L} \]
Constructive analysis of the completeness theorem

LEM $\leftrightarrow \forall \mathcal{T} \varphi. \quad \neg (\mathcal{T} \vdash \varphi) \rightarrow \Gamma \vdash \varphi$

MP $\leftrightarrow \forall \mathcal{T} \varphi. \quad \mathcal{E} \mathcal{T} \rightarrow \neg (\mathcal{T} \vdash \varphi) \rightarrow \mathcal{T} \vdash \varphi$

MP$_L$ $\leftrightarrow \forall \mathcal{T} \varphi. \quad \mathcal{E}_L \mathcal{T} \rightarrow \neg (\mathcal{T} \vdash \varphi) \rightarrow \mathcal{T} \vdash \varphi$

$\leftrightarrow \forall \Gamma \varphi. \quad \neg (\Gamma \not\vdash \varphi) \rightarrow \Gamma \vdash \varphi$
Constructive analysis of the completeness theorem

\[
\text{DNE}(P : \forall X. (X \to \mathbb{P}) \to \mathbb{P}) := \forall Xp. \ PXp \to \forall \ x. \neg \neg px \to px
\]

\[
\text{LEM} \leftrightarrow \forall T \varphi. \ \neg (T \not\vdash \varphi) \to \Gamma \vdash \varphi \leftrightarrow \text{DNE}(\lambda Xp. \top)
\]

\[
\text{MP} \leftrightarrow \forall T \varphi. \ \mathcal{E}T \to \neg (T \not\vdash \varphi) \to T \vdash \varphi \leftrightarrow \text{DNE}(\lambda Xp. \mathcal{D}X \land \mathcal{E} \ p)
\]

\[
\text{MP}_L \leftrightarrow \forall T \varphi. \ \mathcal{E}_L T \to \neg (T \not\vdash \varphi) \to T \vdash \varphi \leftrightarrow \text{DNE}(\lambda Xp. \mathcal{D}_L X \land \mathcal{E}_L p)
\]

\[
\leftrightarrow \forall \Gamma \varphi. \ \neg (\Gamma \not\vdash \varphi) \to \Gamma \vdash \varphi \leftrightarrow \text{DNE}(\lambda Xp. \ p = \lambda s. \exists t.s \triangleright t)
\]
Constructive analysis of the completeness theorem

\[ \text{DNE}(P : \forall X. (X \to P) \to P) := \forall Xp. PXp \to \forall x. \neg \neg px \to px \]

**LEM** \iff \forall T \varphi. \neg(T \not\vdash \varphi) \to \Gamma \vdash \varphi \iff \text{DNE}(\lambda Xp. T)

**MP** \iff \forall T \varphi. ET \to \neg(T \not\vdash \varphi) \to T \vdash \varphi \iff \text{DNE}(\lambda Xp. DX \land E p)

**MP_L** \iff \forall T \varphi. E_L T \to \neg(T \not\vdash \varphi) \to T \vdash \varphi \iff \text{DNE}(\lambda Xp. D_L X \land E_L p)

**D. Kirst., D. Wehr, Y. Forster**

Theorem

Let \( p : X \to P \) and \( q : Y \to P \). If \( p \) is stable and \( p \preceq q \) then \( q \) is stable, where

\[ \exists f : X \to Y. \forall x. px \iff q(fx). \]
Undecidability: The Entscheidungsproblem*

*F., K., and Gert Smolka at CPP’19.
General Idea

Conventional outline following Turing:

- Encode Turing machine $M$ as formula $\varphi_M$ over custom signature
- Verify that $M$ halts if and only if $\varphi_M$ holds in all models
- Verify that $M$ halts if and only if $\varphi_M$ is provable in intuitionistic natural deduction
- Verify that $M$ halts if and only if $\varphi_M$ is provable in classical natural deduction
General Idea

Conventional outline following Turing:

- Encode Turing machine $M$ as formula $\varphi_M$ over custom signature
- Verify that $M$ halts if and only if $\varphi_M$ holds in all models
- Verify that $M$ halts if and only if $\varphi_M$ is provable in intuitionistic natural deduction
- Verify that $M$ halts if and only if $\varphi_M$ is provable in classical natural deduction

We follow the simpler proof due to Floyd given by Manna (2003) based on PCP.
The Post Correspondence Problem PCP*

\[
C_2 \quad xfor \quad nf \quad FLo \quad d \quad 018inO

\]

*Post (1946); Forster et al. (2018)
The Post Correspondence Problem PCP*

Base type: \( L(B \times B) \)

Definition: \( \text{PCP}(L) := \exists x : L \uplus (u, v) \in L \uplus (x, y) (u, v) \in L \uplus (x + u, y + v) \)

Theorem
The halting problem many-one reduces to PCP.

*Post (1946); Forster et al. (2018)
The Post Correspondence Problem PCP*

Theorem

The halting problem many-one reduces to PCP.

*Post (1946); Forster et al. (2018)
The Post Correspondence Problem PCP

**Theorem**

The halting problem many-one reduces to PCP.

*Post (1946); Forster et al. (2018)*
The Post Correspondence Problem PCP*

Theorem

The halting problem many-one reduces to PCP.

*Post (1946); Forster et al. (2018)
The Post Correspondence Problem PCP*

Base type: \( \mathbb{L}(\mathbb{L} \times \mathbb{L}) \)

Definition: PCP\( (\mathbb{L}) := \exists x : \mathbb{L}. \mathbb{L} \triangleleft (x, x) \) \( (u, v) \in \mathbb{L} \triangleleft (x, y) \) \((u, v) \in \mathbb{L} \triangleleft (x + u, y + v) \)

Theorem

The halting problem many-one reduces to PCP.

\*Post (1946); Forster et al. (2018)
The Post Correspondence Problem PCP

<table>
<thead>
<tr>
<th>C2</th>
<th>xfor</th>
<th>nf</th>
<th>FLo</th>
<th>d</th>
<th>018inO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoC2018</td>
<td>——</td>
<td>——</td>
<td>F</td>
<td>ord</td>
<td>inOxf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FLo</th>
<th>C2</th>
<th>018inO</th>
<th>xfor</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>LoC2018</td>
<td>inOxf</td>
<td>——</td>
<td>ord</td>
</tr>
</tbody>
</table>

**Theorem**
The halting problem many-one reduces to PCP.

---

*Post (1946); Forster et al. (2018)*
The Post Correspondence Problem PCP*

Base type: $L(B \times B)$

Definition: $\text{PCP}(L) := \exists x : B. L \triangleright (x, x)$

Theorem

The halting problem many-one reduces to PCP.

*Post (1946); Forster et al. (2018)
A Standard Model

Strings can be encoded as terms, e.g. \( ttff ff tt = f_{tt}(f_{ff}(f_{tt}(e)))) \).
A Standard Model

Strings can be encoded as terms, e.g. \( ttfffftt = f_{tt} \left( f_{ff} \left( f_{tt} \left( e \right) \right) \right) \).

The standard model \( \mathcal{B} \) over the type \( \mathcal{L}(\mathbb{B}) \) of Boolean strings captures exactly the cards derivable from a fixed stack \( S \):

\[
\begin{align*}
\mathit{e}^\mathcal{B} & := [] \\
\mathit{f}_b^\mathcal{B} \ s & := b :: s \\
\mathit{Q}^\mathcal{B} & := \text{PCP} \ S \\
\mathit{P}^\mathcal{B} \ s \ t & := S \triangleright s/t.
\end{align*}
\]
A Standard Model

Strings can be encoded as terms, e.g. $ttfffftt = f_{tt}(f_{ff}(f_{tt}(e))))$.

The standard model $\mathcal{B}$ over the type $\mathcal{L}(\mathbb{B})$ of Boolean strings captures exactly the cards derivable from a fixed stack $S$:

$$
e^\mathcal{B} := []$$

$$f^\mathcal{B}_b s := b :: s$$

$$Q^\mathcal{B} := \text{PCP } S$$

$$P^\mathcal{B} s t := S \triangleright s/t.$$

Lemma

Let $\rho : \mathbb{N} \to \mathcal{L}(\mathbb{B})$ be an environment for the standard model $\mathcal{B}$. Then $\hat{\rho} s = s$ and $\mathcal{B} \models^\rho P \tau_1 \tau_2 \leftrightarrow S \triangleright \hat{\rho} \tau_1 / \hat{\rho} \tau_2$. 
The reduction

We express the constructors of $S \triangleright s/t$ and PCP as formulas:

$\varphi_1 := [P s t | s/t \in S ]$ \hspace{1cm} $\varphi_2 := [ \forall xy. P x y \rightarrow P(sx)(ty) | s/t \in S ]$ \hspace{1cm} $\varphi_3 := \forall x. P x x \rightarrow Q$

$\varphi_S := \varphi_1 \rightarrow \varphi_2 \rightarrow \varphi_3 \rightarrow Q$
The reduction

We express the constructors of $S \triangleright s/t$ and PCP as formulas:

$$\varphi_1 := [P \bar{s} \bar{t} \mid s/t \in S] \quad \varphi_2 := [\forall xy. \, P x y \rightarrow P (\bar{s}x)(\bar{t}y) \mid s/t \in S] \quad \varphi_3 := \forall x. \, P x x \rightarrow Q$$

$$\varphi_S := \varphi_1 \rightarrow \varphi_2 \rightarrow \varphi_3 \rightarrow Q$$

1. $\text{PCP } S \rightarrow \vdash \varphi_S$
2. $\vdash$ is sound for Tarski semantics w.r.t. all models
3. $B \models \varphi_S \rightarrow \text{PCP } S$

**Theorem**

PCP reduces to Tarski validity (w.r.t. all models) and intuitionistic provability.

**Theorem**

PCP reduces to Tarski satisfiability (w.r.t. any model).
Undecidability of Classical Provability

Soundness is not usable!
Undecidability of Classical Provability

Soundness is not usable! As a remedy, we define a Gödel-Gentzen-Friedman translation $\varphi^Q$ of formulas $\varphi$ such that $A \vdash c \varphi$ implies $A^Q \vdash \varphi^Q$. 

1. $\forall \Gamma \varphi. \Gamma \vdash \varphi \rightarrow \Gamma \vdash c \varphi$

2. $B \models \varphi^Q \rightarrow B \models \varphi$

Theorem PCP$^S$ iff $\vdash c \varphi^S$, hence PCP reduces to classical provability.
Undecidability of Classical Provability

Soundness is not usable! As a remedy, we define a Gödel-Gentzen-Friedman translation $\varphi^Q$ of formulas $\varphi$ such that $A \vdash_c \varphi$ implies $A^Q \vdash \varphi^Q$.

1. $\forall \Gamma \varphi. \Gamma \vdash \varphi \to \Gamma \vdash_c \varphi$
2. $B \models \varphi^Q_S \to B \models \varphi_S$

Theorem

PCP $S$ iff $\vdash_c \varphi_S$, hence PCP reduces to classical provability.
Recipe for undecidability proofs

1. The halting problem is undecidable.
2. The halting problem reduces to PCP.
3. The reduction function is computable.
4. PCP reduces to FOL.
5. The reduction function is computable.
6. Computable reductions transport undecidability.
Recipe for undecidability proofs

1. The halting problem is undecidable.
2. The halting problem reduces to PCP.
3. The reduction function is computable.
4. PCP reduces to FOL.
5. The reduction function is computable.
6. Computable reductions transport undecidability.
Theorem V  For every \( m,n \geq 1 \), there exists a recursive function \( s_n^m \) of \( m + 1 \) variables such that for all \( x, y_1, \ldots, y_m \),

\[
\lambda z_1 \cdots z_n [\varphi_x^{(m+n)}(y_1, \ldots, y_m, z_1, \ldots, z_n)] = \varphi_{s_n^m(x,y_1,\ldots,y_m)}^{(n)}.
\]

Proof. Take the case \( m = n = 1 \). (Proof is analogous for the other cases.) Consider the family of all partial functions of one variable which are expressible as \( \lambda z[\varphi_x^{(2)}(y,z)] \) for various \( x \) and \( y \). Using our standard formal characterization for functions of two variables, we can view this as a new formal characterization for a class of partial recursive functions of one variable. By Part III of the Basic Result, there exists a uniform effective procedure for going from sets of instructions in this new characterization to sets of instructions in the old. Hence, by Church's Thesis, there must be a recursive function \( f \) of two variables such that

\[
\lambda z[\varphi_x^{(2)}(y,z)] = \varphi_{f(x,y)}.
\]

This \( f \) is our desired \( s_1^1 \). \( \Box \)

The informal argument by appeal to Church's Thesis and Part III of the Basic Result can be replaced by a formal proof. (Indeed, the functions \( s_n^m \) can be shown to be primitive recursive.) We refer the reader to Davis [1958] and Kleene [1952]. Theorem V is known as the \( s \)-\( m \)-\( n \) theorem and is due to Kleene. Theorem V (together with Church's Thesis) is a tool of great range and power.
Theorem 1.1. There is a primitive recursive function \( g, \phi, \delta \) such that, for all \( n \in \mathbb{N} \),
\[
\mu = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta < n \Rightarrow \phi(n, \eta) = g(\nu, \eta)(\delta(\nu, \eta)) \}
\]

Intuitively, the result may be interpreted, for \( f \equiv g, \phi \equiv \delta \), that the author of any sufficiently nice conclusion, \( f \), gives any Turing machine \( M \) and number \( n \) a Turing machine \( M' \) can be found such that
\[
\mu = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta < n \Rightarrow M'(n, \eta) = f(\nu, \eta)(\delta(\nu, \eta)) \}
\]

Now it is clear that there exist Turing machines \( M_n \) satisfying this last condition, for all natural numbers \( n \).\( M_n \) is a suitable partial recursive function of \( \mu \). Hence, the notion of an \( n \)-membered Turing machine is that of a Turing machine \( M_n \) which, for all \( n \)-element sets \( D \) of alphanumeric symbols, is to be described as a Turing machine which, having as \( n \) \# of points in the set \( D \), eventually prints \( g(\mu, D)(\delta(\mu, D)) \) and then proceeds to act like \( f \) when confronted

Initially, as inspection proves, to a random recursive function \( g, \phi, \delta \). As the general case does not differ essentially from this special case, all that is required for a formal proof is a detailed conclusion of \( M_n \), and a careful presentation of the odd numbers. The reader who wishes to omit the tedious details, and simply accept the result, may well do so.

Theorem 1.2. For every value of \( n \), let \( F_n \) be the Turing machine consisting of the following quads:
\[
\quad L P \quad L F \quad L S \quad L B \\
\quad \vdots \quad \vdots \quad \vdots \quad \vdots
\]

Then, with respect to \( F_n \),
\[
\mu = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta \leq n \Rightarrow F_n^*(n, \eta) = g(\nu, \eta)(\delta(\nu, \eta)) \}
\]

Let \( x \) be a Godel number of a Turing machine \( M \), and let
\[
F_n^*(n, \eta) \equiv x \quad \forall \eta \leq n
\]

Then, since the quadruples of \( F_n \) have precisely the same effect on \( M^* \), we have
\[
\mu = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta \leq n \Rightarrow F_n^*(n, \eta) = g(\nu, \eta)(\delta(\nu, \eta)) \}
\]

We prove now to evaluate one of the Godel numbers of \( F_n \) as a function of \( r \) and \( \phi \). The Godel number of the quadruple that maps \( r \) and \( \phi \), and
\[
\mu = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta \leq n \Rightarrow F_n^*(n, \eta) = g(\nu, \eta)(\delta(\nu, \eta)) \}
\]

Then, if we set
\[
\mu = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta \leq n \Rightarrow F_n^*(n, \eta) = g(\nu, \eta)(\delta(\nu, \eta)) \}
\]

then \( \mu \) is a primitive recursive function, and, for each \( n \), \( \mu(n) \) is a Godel number of \( F_n \).

We recall that the predicate \( \mathcal{E}(\mu) \), which is true if and only if \( \mu \) is the number associated with an external configuration, is primitive recursive, since
\[
\mathcal{E}(\mu) = \bigvee \{ \mathcal{E}(n) \mid n \leq \mu \}
\]

Hence, the function \( \mathcal{E}(\mu) \), which is \( 1 \) when \( \mu \) is the number associated with a configuration, is primitive recursive. Let \( \mathcal{O}(\mu) \) be the Godel number of the quadruple that maps \( \mathcal{O}(\mu) \) and \( \mu \), and the number of the quadruple obtained from the one by replacing each \( \mu \) by \( \mathcal{O}(\mu) \).

\[
\mathcal{O}(\mu) = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta < n \Rightarrow \phi(n, \eta) = \mathcal{O}(\nu, \eta)(\delta(\nu, \eta)) \}
\]

and if \( \mathcal{O}(\mu) \) is primitive recursive, hence, if we set
\[
\mathcal{O}(\mu) = \bigvee \{ \mathcal{O}(n) \mid n \leq \mu \}
\]

then \( \mathcal{O}(\mu) \) is a primitive recursive function, and, for each \( n \), \( \mathcal{O}(n) \) is a Godel number of \( \mathcal{O}(n) \).

Let \( \mathcal{O}(\mu) \) be the Godel number of a Turing machine \( \mathcal{O}(\mu) \), otherwise.

Then, by (11) of Chapter 4, Sec. 3, \( \mathcal{O}(\mu) \) is primitive recursive. Finally, let
\[
\mu = \min \{ \nu \mid \text{for all } x, \forall \eta \leq x, \eta < n \Rightarrow \phi(n, \eta) = \mathcal{O}(\nu, \eta)(\delta(\nu, \eta)) \}
\]

Then, \( \mu \) is a primitive recursive function, and, for each \( n \), \( \mu(n) \) is a Godel number of \( \mathcal{O}(\mu) \), otherwise.
Important intermediate results with E. Heiter, D. Larchey-Wendling, F. Kunze, G. Smolka, M. Wuttke, M. Roth at ITP '18, CPP '19, ITP '19, POPL '20, ITP '21
Synthetic Undecidability and the weak call-by-value λ-calculus L

Isolate the weak call-by-value λ-calculus as central model

- Turing-complete model of computation with reasonable time and space measures
- Extraction framework from fragment of Coq to L (F. and Fabian Kunze at ITP ’19) allows relatively easy programming in L
- Define undecidability as
  \[ U(p) := \mathcal{D}p \rightarrow \mathcal{E}(\text{Halt}) \]

- We can prove usual undecidability by extracting reduction functions to L
- We can develop synthetic computability theory based on axiom CT_L stating that
  \[ \forall f : \mathbb{N} \rightarrow \mathbb{N}. \exists t. \text{ the L-term } t \text{ computes } f \]

*Kreisel (1965); Forster (2021, 2022)
More Constructive Reverse Mathematics *

Diener (2020) (in Bishop style constructive math):
Compactness is equivalent to Weak König’s Lemma for decidable trees $\text{WKL}_D$
$\text{WKL}_D$ is equivalent to the fan theorem $\text{FAN}_D$

Simpson (1985) (in classical reverse mathematics, $\text{RCA}_0$):
The model existence theorem is equivalent to $\text{WKL}$

Completeness of intuitionistic FOL w.r.t. Kripke semantics is equivalent to $\text{FAN}_D$

The problem is countable/unique choice. Without countable/unique choice, total relations $\mathbb{N} \to \mathbb{B} \to \mathbb{P}$ and functions $\mathbb{N} \to \mathbb{B}$ are not the same objects

We then have (Berger et al. (2012))

$$\text{WKL} \leftrightarrow \text{LLPO} \land \Pi^0_1-\text{AC}_{\mathbb{N},\mathbb{B}}$$
Models matter

WKL := Every infinite binary tree has an infinite path
Models matter

WKL := Every infinite binary tree has an infinite path (a boolean function!)
Models matter

WKL := Every infinite binary tree has an infinite path (a boolean function!)

A model $\mathcal{M}$ is **decidable** if $D(\lambda P. \hat{P})$ (predicate interpretations are boolean functions).

A model $\mathcal{M}$ is **omniscient** if $D(\lambda \rho \varphi. \mathcal{M} \vdash_\rho \varphi)$ (everything is a boolean function).
Models matter

\[ \text{WKL} := \text{Every infinite binary tree has an infinite path (a boolean function!)} \]

A model \( \mathcal{M} \) is decidable if \( D(\lambda P, \hat{P}v) \) (predicate interpretations are boolean functions).

A model \( \mathcal{M} \) is omniscient if \( D(\lambda \rho \varphi. \mathcal{M} \vdash_{\rho} \varphi) \) (everything is a boolean function).

**Theorem**

The following are equivalent:

1. Completeness of \( T \vdash_{c} \varphi \) for omniscient/decidable models.
2. LEM and model existence for omniscient/decidable models.
3. LEM and compactness for omniscient/decidable models.
4. LEM and WKL.
5. Every predicate \( \mathbb{N} \rightarrow \mathbb{P} \) is decidable.
Models matter

WKL := Every infinite binary tree has an infinite path (a boolean function!)
A model $\mathcal{M}$ is decidable if $D(\lambda P. \hat{P})$ (predicate interpretations are boolean functions).
A model $\mathcal{M}$ is omniscient if $D(\lambda \rho \varphi. \mathcal{M} \vdash \rho \varphi)$ (everything is a boolean function).

Theorem

The following are equivalent:

1. Completeness of $T \vdash_c \varphi$ for omniscient/decidable models.
2. LEM and model existence for omniscient/decidable models.
3. LEM and compactness for omniscient/decidable models.
4. LEM and WKL.
5. Every predicate $\mathbb{N} \to \mathbb{P}$ is decidable.

WKL$_D$ := Every decidable and infinite binary tree has an infinite path

Corollary

Compactness for decidable models implies WKL$_D$. 
Open questions

- What happens if we restrict to enumerable / finite theories?
- Can we prove equivalences for $\text{WKL}_D$?
- Is there a uniform theorem with $\text{DNE}(P) \land \text{WKL}_P$?
- What’s the status of Kripke completeness?
  - Does WKL always play a role, or just for decidable models?
  - If we add $\exists$ and $\lor$, does this change the necessary axioms?

\[
\text{WKL}_D \iff \forall R : \mathbb{N} \rightarrow \mathbb{B} \rightarrow \mathbb{P}. \ \Pi^0_1 R \rightarrow (\forall n. \neg\neg \exists b. Rnb) \rightarrow \exists f : \mathbb{N} \rightarrow \mathbb{B}. \forall n. R (fn)
\]
Undecidability: Trakhtenbrot’s Theorem*

General idea

Given a FOL formula $\varphi$, is $\varphi$ finitely satisfiable?

Textbook proofs by dual reduction from the halting problem:

- Encode Turing machine $M$ as formula $\varphi_M$ over custom signature
- Verify that the models of $\varphi_M$ correspond to the runs of $M$
- Conclude that $M$ halts if and only if $\varphi_M$ has a finite model

*e.g. Libkin (2010); Börger et al. (1997)*
Given a FOL formula $\varphi$, is $\varphi$ finitely satisfiable?

Textbook proofs by dual reduction from the halting problem:

- Encode Turing machine $M$ as formula $\varphi_M$ over custom signature
- Verify that the models of $\varphi_M$ correspond to the runs of $M$
- Conclude that $M$ halts if and only if $\varphi_M$ has a finite model

Our mechanisation:

- Illustrates that one can still use PCP for a simpler reduction
- Signature minimisations are constructive for finite models

* e.g. Libkin (2010); Börger et al. (1997)
Finiteness in Constructive Type Theory

Definition

A type $X$ is **finite** if there exists a list $l_X$ with $x \in l_X$ for all $x : X$. 
Finiteness in Constructive Type Theory

**Definition**

A type $X$ is **finite** if there exists a list $l_X$ with $x \in l_X$ for all $x : X$.

This seems to be a good compromise:
Finiteness in Constructive Type Theory

**Definition**

A type $X$ is **finite** if there exists a list $l_X$ with $x \in l_X$ for all $x : X$.

This seems to be a good compromise:

- Easy to establish and work with
Finiteness in Constructive Type Theory

**Definition**

A type \( X \) is finite if there exists a list \( l_X \) with \( x \in l_X \) for all \( x : X \).

This seems to be a good compromise:

- Easy to establish and work with
- Does not enforce discreteness
Finiteness in Constructive Type Theory

**Definition**

A type $X$ is **finite** if there exists a list $l_X$ with $x \in l_X$ for all $x : X$.

This seems to be a good compromise:

- Easy to establish and work with
- Does not enforce discreteness
- Enough to get expected properties:
  - Every strict order on a finite type is well-founded
  - Every finite decidable equivalence relation admits a quotient on $\mathbb{F}_n$
Finiteness in Constructive Type Theory

Definition

A type $X$ is finite if there exists a list $l_X$ with $x \in l_X$ for all $x : X$.

This seems to be a good compromise:

- Easy to establish and work with
- Does not enforce discreteness
- Enough to get expected properties:
  - Every strict order on a finite type is well-founded
  - Every finite decidable equivalence relation admits a quotient on $\mathbb{F}_n$

\[ FSAT(\Sigma) \varphi \] if additionally $D$ is finite and all $P^M$ are decidable

\[ FSATEQ(\Sigma; \equiv) \varphi \] if $x \equiv^M y \leftrightarrow x = y$ for all $x, y : D$ (hence discrete)
Encoding the Post Correspondence Problem

We use the signature \( \Sigma_{\text{BPCP}} := (\{^0, e^0, f_{tt}^1, f_{ff}^1\}; \{P^2, \prec^2, \equiv^2\}) \):

Chains like \( f_{ff}(f_{tt}(e)) \) represent strings while \( ^0 \) signals overflow. \( P \) concerns only defined values and \( \prec \) is a strict ordering:

\[
\phi_P := \forall xy. P x y \rightarrow x \not\equiv ^0 \land y \not\equiv ^0
\]

\[
\phi_\prec := (\forall x. x \not\prec x) \land (\forall xyz. x \prec y \rightarrow y \prec z \rightarrow x \prec z)
\]

Sanity checks on \( f \) regarding overflow, disjointness, and injectivity:

\[
\phi_f := \begin{cases}
  f_{tt}^1 e^0 \equiv^0 \equiv^0 \land f_{ff}^1 e^0 \equiv^0 \equiv^0 \\
  \forall x. f_{tt}^1 x \not\equiv e^0 \land f_{ff}^1 x \not\equiv e^0 \\
  \forall xy. f_{tt}^1 x \not\equiv ^0 \rightarrow f_{tt}^1 x \equiv f_{tt}^1 y \rightarrow x \equiv y \\
  \forall xy. f_{ff}^1 x \not\equiv ^0 \rightarrow f_{ff}^1 x \equiv f_{ff}^1 y \rightarrow x \equiv y \\
  f_{tt}^1 x \equiv f_{ff}^1 y \rightarrow f_{tt}^1 x \equiv ^0 \land f_{ff}^1 y \equiv ^0
\end{cases}
\]
Encoding the Post Correspondence Problem

We use the signature $\Sigma_{BPCP} := (\{\star^0, e^0, f^1_{tt}, f^1_{ff}\}; \{P^2, \preceq^2, \equiv^2\})$:

- Chains like $f_{ff}(f_{tt}(e))$ represent strings while $\star$ signals overflow
Encoding the Post Correspondence Problem

We use the signature $\Sigma_{\text{BPCP}} := (\{\star^0, e^0, f_{tt}^1, f_{ff}^1\}; \{P^2, \prec^2, \equiv^2\}):$

- Chains like $f_{ff}(f_{tt}(e))$ represent strings while $\star$ signals overflow
- $P$ concerns only defined values and $\prec$ is a strict ordering:

\[
\begin{align*}
\varphi_P & := \forall xy. P x y \implies x \neq \star \land y \neq \star \\
\varphi_\prec & := (\forall x. x \neq x) \land (\forall xyz. x \prec y \implies y \prec z \implies x \prec z)
\end{align*}
\]
Encoding the Post Correspondence Problem

We use the signature $\Sigma_{BPCP} := (\{ \star^0, e^0, f^1_{tt}, f^1_{ff}\}; \{ P^2, <^2, \equiv^2 \})$:

- Chains like $f_{ff}(f_{tt}(e))$ represent strings while $\star$ signals overflow
- $P$ concerns only defined values and $<$ is a strict ordering:

  \[
  \varphi_P := \forall xy. P x y \rightarrow x \not\equiv \star \land y \not\equiv \star \\
  \varphi_< := (\exists x. x \not< x) \land (\forall xyz. x < y \rightarrow y < z \rightarrow x < z)
  \]

- Sanity checks on $f$ regarding overflow, disjointness, and injectivity:

  \[
  \varphi_f := \left( f_{tt} \star \equiv \star \land f_{ff} \star \equiv \star \right) \land \left( \forall x. f_{tt} x \not\equiv e \land f_{ff} x \not\equiv e \right)
  \]

D. Kirst., D. Wehr, Y. Forster
Analysing FOL in CTT
December 16th/17th, 2021
Trakhtenbrot’s Theorem

Given an instance $R$ of PCP, we construct a formula $\varphi_R$ by:

$$\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_\triangleright \land \exists x. P x x$$
Trakhtenbrot’s Theorem
Given an instance \( R \) of PCP, we construct a formula \( \varphi_R \) by:

\[
\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_\triangleright \land \exists x. P \times x
\]

Crucially, we enforce that \( P \) satisfies the inversion principle of \( R \triangleright (s, t) \):

\[
\varphi_\triangleright := \forall xy. P \times y \rightarrow \bigvee_{(s,t) \in R} \exists uv. P \times v \land x \equiv s \land y \equiv t \land \exists u \land y \equiv t \land u/v < x/y
\]
Trakhtenbrot’s Theorem

Given an instance $R$ of PCP, we construct a formula $\varphi_R$ by:

$$
\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_\triangleright \land \exists x. P x x
$$

Crucially, we enforce that $P$ satisfies the inversion principle of $R \triangleright (s, t)$:

$$
\varphi_\triangleright := \forall xy. P x y \rightarrow \bigvee_{(s, t) \in R} \bigvee \{ x \equiv s \land y \equiv t \land \exists uv. P u v \land x \equiv s u \land y \equiv t v \land u/v < x/y \}
$$

Theorem

PCP $R$ iff FSATEQ($\Sigma_{\text{BPCP}}; \equiv$)$\varphi_R$, hence PCP $\preceq$ FSATEQ($\Sigma_{\text{BPCP}}; \equiv$).
Trakhtenbrot’s Theorem

Given an instance $R$ of PCP, we construct a formula $\varphi_R$ by:

$$
\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_\triangleright \land \exists x. P x x
$$

Crucially, we enforce that $P$ satisfies the inversion principle of $R \triangleright (s, t)$:

$$
\varphi_\triangleright := \forall xy. P x y \rightarrow \bigvee_{(s,t) \in R} \exists uv. P u v \land x \equiv s \land y \equiv t
$$

Theorem

PCP $R$ iff FSATEQ($\Sigma_{BPCP}; \equiv$)$\varphi_R$, hence PCP $\preceq$ FSATEQ($\Sigma_{BPCP}; \equiv$).

Proof.
Trakhtenbrot’s Theorem

Given an instance $R$ of PCP, we construct a formula $\varphi_R$ by:

$$\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_{\triangleright} \land \exists x. \ P x x$$

Crucially, we enforce that $P$ satisfies the inversion principle of $R \triangleright (s, t)$:

$$\varphi_{\triangleright} := \forall xy. \ P x y \rightarrow \bigvee_{(s,t) \in R} \exists uv. \ P u v \land x \equiv \bar{s} \land y \equiv \bar{t} \land \exists u v. \ P u v \land x \equiv \bar{s} u \land y \equiv \bar{t} v \land u/v < x/y$$

**Theorem**

PCP $R$ iff $\text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv) \varphi_R$, hence $\text{PCP} \preceq \text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv)$.

**Proof.**

If $R$ has a solution of length $n$, then $\varphi_R$ is satisfied by the model of strings of length bounded by $n$. 
Trakhtenbrot’s Theorem

Given an instance $R$ of PCP, we construct a formula $\varphi_R$ by:

$$\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_\triangleright \land \exists x. P x x$$

Crucially, we enforce that $P$ satisfies the inversion principle of $R \triangleright (s, t)$:

$$\varphi_\triangleright := \forall xy. P x y \rightarrow \bigvee_{(s, t) \in R} \exists u v. P u v \land x \equiv s \land y \equiv t \land u \equiv s u \land v \equiv t v \land u / v < x / y$$

**Theorem**

PCP $R$ iff FSATEQ$(\Sigma_{\text{BPCP}}; \equiv) \varphi_R$, hence PCP $\preceq$ FSATEQ$(\Sigma_{\text{BPCP}}; \equiv)$.

**Proof.**

If $R$ has a solution of length $n$, then $\varphi_R$ is satisfied by the model of strings of length bounded by $n$. Conversely, if $\mathcal{M} \models_{\rho} \varphi_R$ we can extract a solution of $R$ from $\varphi_\triangleright$ by well-founded induction on $\prec^\mathcal{M}$.
Trakhtenbrot’s Theorem

Given an instance $R$ of PCP, we construct a formula $\varphi_R$ by:

$$\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_\triangleright \land \exists x. P x x$$

Crucially, we enforce that $P$ satisfies the inversion principle of $R \triangleright (s, t)$:

$$\varphi_\triangleright := \forall x y. P x y \rightarrow \forall (s, t) \in R \bigvee \left\{ x \equiv \bar{s} \land y \equiv \bar{t} \land \exists u v. P u v \land x \equiv \bar{s} u \land y \equiv \bar{t} v \land u/v \prec x/y \right\}$$

Theorem

PCP $R$ iff FSATEQ($\Sigma_{BPCP}; \equiv$)$\varphi_R$, hence PCP $\preceq$ FSATEQ($\Sigma_{BPCP}; \equiv$).

Proof.

If $R$ has a solution of length $n$, then $\varphi_R$ is satisfied by the model of strings of length bounded by $n$. Conversely, if $M \models_\rho \varphi_R$ we can extract a solution of $R$ from $\varphi_\triangleright$ by well-founded induction on $\prec_M$ (which is applicable since $M$ is finite).
Signature Transformations

Given a finite and discrete signature $\Sigma$ with arities bounded by $n$, we have:

$$\text{FSATEQ}(\Sigma; \equiv) \preceq \text{FSAT}(\Sigma) \preceq \text{FSAT}(\emptyset; P^{n+2}) \preceq \text{FSAT}(\emptyset; \in^2)$$
Signature Transformations

Given a finite and discrete signature $\Sigma$ with arities bounded by $n$, we have:

$$\text{FSATEQ}(\Sigma; \equiv) \preceq \text{FSAT}(\Sigma) \preceq \text{FSAT}(\emptyset; P^{n+2}) \preceq \text{FSAT}(\emptyset; \in^2)$$

First reduction: axiomatise that $\equiv$ is a congruence for the symbols in $\Sigma$
Signature Transformations

Given a finite and discrete signature $\Sigma$ with arities bounded by $n$, we have:

$$\text{FSATEQ}(\Sigma; \equiv) \preceq \text{FSAT}(\Sigma) \preceq \text{FSAT}(\emptyset; P^{n+2}) \preceq \text{FSAT}(\emptyset; \in^2)$$

First reduction: axiomatise that $\equiv$ is a congruence for the symbols in $\Sigma$

Second reduction:

- Encode $k$-ary functions as $(k + 1)$-ary relations
- Align the relation arities to be constantly $n + 1$
- Merge relations into a single $(n + 2)$-ary relation indexed by constants
- Interpret constants with fresh variables
Signature Transformations

Given a finite and discrete signature $\Sigma$ with arities bounded by $n$, we have:

$$\text{FSATEQ}(\Sigma; \equiv) \preceq \text{FSAT}(\Sigma) \preceq \text{FSAT}(\emptyset; P^{n+2}) \preceq \text{FSAT}(\emptyset; \in^2)$$

First reduction: axiomatise that $\equiv$ is a congruence for the symbols in $\Sigma$

Second reduction:
- Encode $k$-ary functions as $(k + 1)$-ary relations
- Align the relation arities to be constantly $n + 1$
- Merge relations into a single $(n + 2)$-ary relation indexed by constants
- Interpret constants with fresh variables

Caveat: intermediate reductions may rely on discrete models...
Discrete Models

$\text{FSAT}'(\Sigma) \varphi$ if $\text{FSAT}(\Sigma) \varphi$ on a discrete model
Discrete Models

FSAT′(Σ) ϕ if FSAT(Σ) ϕ on a discrete model

Can every finite model M be transformed to a discrete finite model M′?
FSAT′(Σ) ϕ if FSAT(Σ) ϕ on a discrete model

Can every finite model \( M \) be transformed to a discrete finite model \( M' \)?

Idea: first-order indistinguishability \( x \Downarrow y := \forall \varphi. M \models_{x, \rho} \varphi \iff M \models_{y, \rho} \varphi \)
Discrete Models

Can every finite model $\mathcal{M}$ be transformed to a discrete finite model $\mathcal{M}'$?

Idea: first-order indistinguishability $\dot{x} = y \eqdef \forall \rho. \mathcal{M} \models _{x, \rho} \varphi \iff \mathcal{M} \models _{y, \rho} \varphi$

**Lemma**

The relation $\dot{x} = y$ is a decidable congruence for the symbols in $\Sigma$. 

"FSAT'(\Sigma) \varphi" if FSAT(\Sigma) \varphi on a discrete model"
Can every finite model $\mathcal{M}$ be transformed to a discrete finite model $\mathcal{M}'$?

Idea: first-order indistinguishability $x \equiv y := \forall \varphi. \mathcal{M} \models x \cdot \varphi \iff \mathcal{M} \models y \cdot \varphi$

**Lemma**

\textit{The relation} $x \equiv y$ \textit{is a decidable congruence for the symbols in} $\Sigma$.

**Fact**

$\text{FSAT}'(\Sigma) \varphi$ \textit{iff} $\text{FSAT}(\Sigma) \varphi$, hence in particular $\text{FSAT}'(\Sigma) \varphi \preceq \text{FSAT}(\Sigma) \varphi$. 
Discrete Models

Can every finite model $\mathcal{M}$ be transformed to a discrete finite model $\mathcal{M}'$?

Idea: first-order indistinguishability $x \equiv y : \forall \varphi. \mathcal{M} \models_{x \cdot \rho} \varphi \iff \mathcal{M} \models_{y \cdot \rho} \varphi$

Lemma

The relation $x \equiv y$ is a decidable congruence for the symbols in $\Sigma$.

Fact

$FSAT'(\Sigma) \varphi$ iff $FSAT(\Sigma) \varphi$, hence in particular $FSAT'(\Sigma) \varphi \preceq FSAT(\Sigma) \varphi$.

Proof.

If $\mathcal{M} \models_{\rho} \varphi$ pick $\mathcal{M}'$ to be the quotient of $\mathcal{M}$ under $x \equiv y$. \qed
Compressing Relations: $FSAT(\emptyset; P^n) \preceq FSAT(\emptyset; \in^2)$

Intuition: encode $P x_1 \ldots x_n$ as $(x_1, \ldots, x_n) \in p$ for a set $p$ representing $P$
Compressing Relations: $\text{FSAT}(\emptyset; P^n) \preceq \text{FSAT}(\emptyset; \in^2)$

Intuition: encode $P x_1 \ldots x_n$ as $(x_1, \ldots, x_n) \in p$ for a set $p$ representing $P$

So let’s play set theory!
Compressing Relations: $\text{FSAT}(\emptyset; P^n) \preceq \text{FSAT}(\emptyset; \in^2)$

Intuition: encode $P x_1 \ldots x_n$ as $(x_1, \ldots, x_n) \in p$ for a set $p$ representing $P$

So let’s play set theory! For a set $d$ representing the domain we define $\varphi'_\in$:

\[
(P x_1 \ldots x_n)'_\in := "(x_1, \ldots, x_n) \in p"
\]
\[
(\varphi \mathrel{\square} \psi)'_\in := (\varphi)'_\in \mathrel{\square} (\psi)'_\in
\]
\[
(\forall z. \varphi)'_\in := \forall z. z \in d \rightarrow (\varphi)'_\in
\]
\[
(\exists z. \varphi)'_\in := \exists z. z \in d \land (\varphi)'_\in
\]
Compressing Relations: $\text{FSAT}({\emptyset}; P^n) \preceq \text{FSAT}({\emptyset}; \in^2)$

Intuition: encode $P \times_1 \ldots \times_n$ as $(x_1, \ldots, x_n) \in p$ for a set $p$ representing $P$.

So let’s play set theory! For a set $d$ representing the domain we define $\varphi'_\in$:

$$(P \times_1 \ldots \times_n)'\in : = "(x_1, \ldots, x_n) \in p" \quad (\forall z. \varphi)'\in : = \forall z. z \in d \rightarrow (\varphi)'\in$$

$$(\varphi \square \psi)'\in : = (\varphi)'\in \square (\psi)'\in \quad (\exists z. \varphi)'\in : = \exists z. z \in d \land (\varphi)'\in$$

Then $\varphi_\in$ is $\varphi'_\in$ plus asserting $\in$ to be extensional and $d$ to be non-empty.
Compressing Relations: \( \text{FSAT}(\emptyset; P^n) \leq \text{FSAT}(\emptyset; \in^2) \)

Intuition: encode \( P \) as \((x_1, \ldots, x_n) \in p\) for a set \( p \) representing \( P \).

So let’s play set theory! For a set \( d \) representing the domain we define \( \varphi_\in' \):

\[
(P x_1 \ldots x_n)_\in' := "(x_1, \ldots, x_n) \in p" \\
(\forall z. \varphi)_\in' := \forall z. z \in d \rightarrow (\varphi)_\in'
\]

Then \( \varphi_\in \) is \( \varphi_\in' \) plus asserting \( \in \) to be extensional and \( d \) to be non-empty.

**Fact**

\( \text{FSAT}(\emptyset; P^n) \varphi \iff \text{FSAT}(\emptyset; \in^2) \varphi_\in \), hence \( \text{FSAT}(\emptyset; P^n) \preceq \text{FSAT}(\emptyset; \in^2) \).
Compressing Relations: $\text{FSAT}(\emptyset; P^n) \preceq \text{FSAT}(\emptyset; \in^2)$

Intuition: encode $P \times_1 \ldots \times_n$ as $(x_1, \ldots, x_n) \in p$ for a set $p$ representing $P$

So let’s play set theory! For a set $d$ representing the domain we define $\varphi'_\in$: 

$$(P \times_1 \ldots \times_n)'_\in := "(x_1, \ldots, x_n) \in p"$$

$$(\forall z. \varphi)'_\in := \forall z. z \in d \rightarrow (\varphi)'_\in$$

$$(\varphi \Box \psi)'_\in := (\varphi)'_\in \Box (\psi)'_\in$$

$$(\exists z. \varphi)'_\in := \exists z. z \in d \land (\varphi)'_\in$$

Then $\varphi_\in$ is $\varphi'_\in$ plus asserting $\in$ to be extensional and $d$ to be non-empty.

Fact

$\text{FSAT}(\emptyset; P^n) \varphi$ iff $\text{FSAT}(\emptyset; \in^2) \varphi_\in$, hence $\text{FSAT}(\emptyset; P^n) \preceq \text{FSAT}(\emptyset; \in^2)$.

Proof.

The hard direction is to construct a model of $\varphi_\in$ given a model $\mathcal{M}$ of $\varphi$. We employ a segment of the model of hereditarily finite sets by Smolka and Stark (2016) large enough to accommodate $\mathcal{M}$.

□
Full Signature Classification

Composing all signature transformations verified we obtain:

**Theorem**

If $\Sigma$ contains either an at least binary relation or a unary relation together with an at least binary function, then PCP reduces to FSAT($\Sigma$).

On the other hand, FSAT for monadic signatures remains decidable:

**Theorem**

If $\Sigma$ is discrete and has all arities bounded by 1 or if all relation symbols have arity 0, then FSAT($\Sigma$) is decidable.

In any case, since one can enumerate all finite models up to extensionality:

**Fact**

If $\Sigma$ is discrete and enumerable, then FSAT($\Sigma$) is enumerable.
Full Signature Classification

Composing all signature transformations verified we obtain:

**Theorem**

If $\Sigma$ contains either an at least binary relation or a unary relation together with an at least binary function, then PCP reduces to FSAT($\Sigma$).

On the other hand, FSAT for monadic signatures remains decidable:

**Theorem**

If $\Sigma$ is discrete and has all arities bounded by 1 or if all relation symbols have arity 0, then FSAT($\Sigma$) is decidable.
Full Signature Classification

Composing all signature transformations verified we obtain:

**Theorem**

If $\Sigma$ contains either an at least binary relation or a unary relation together with an at least binary function, then PCP reduces to $\text{FSAT}(\Sigma)$.

On the other hand, FSAT for monadic signatures remains decidable:

**Theorem**

If $\Sigma$ is discrete and has all arities bounded by 1 or if all relation symbols have arity 0, then $\text{FSAT}(\Sigma)$ is decidable.

In any case, since one can enumerate all finite models up to extensionality:

**Fact**

If $\Sigma$ is discrete and enumerable, then $\text{FSAT}(\Sigma)$ is enumerable.
Relativised Entscheidungsproblem and Incompleteness∗

∗K. and Marc Hermes at ITP’21.
General Idea

Relativised Entscheidungsproblem: is a formula $\varphi$ entailed by an axiomatisation $A$?

Strategy if $A$ is strong enough to capture computation:

- Encode Turing machine $M$ as formula $\varphi_M$
- Verify that $M$ halts iff $A \models \varphi_M$
- Verify that $M$ halts iff $A \vdash \varphi_M$ (→ direction by hand)
- Instead of TM use problems suitable to encode in $A$
General Idea

Relativised Entscheidungsproblem: is a formula $\varphi$ entailed by an axiomatisation $A$?

Strategy if $A$ is strong enough to capture computation:
- Encode Turing machine $M$ as formula $\varphi_M$
- Verify that $M$ halts iff $A \vdash \varphi_M$
- Verify that $M$ halts iff $A \vdash \varphi_M$ (→ direction by hand)
- Instead of TM use problems suitable to encode in $A$

Undecidability of $A$ implies consistency and incompleteness:
- Reducing a non-trivial problem $P$ to $A \vdash \varphi$ shows $A$ consistent
- Undecidability implies incompleteness for enumerable axiomatisations
Connection of Undecidability to Consistency and Incompleteness

Fact (Consistency)

If \( p \leq A^\vdash \) and there is \( x \) with \( \neg p \times \) then \( A \not\vdash \bot \).

Fact (Synthetic Incompleteness)

If \( A \) is complete (\( \forall \phi. A \vdash \phi \lor A \vdash \neg \phi \)) and consistent, then \( A \vdash \) is decidable.

Proof.

\( A \vdash \) is enumerable and, given completeness and consistency, also co-enumerable as then \( A \not\vdash \phi \iff A \vdash \neg \phi \).

Classically, this is enough to deduce decidability, in our case we need to first observe that \( A \vdash \) is definite, i.e. that \( A \vdash \phi \lor A \not\vdash \phi \).
Connection of Undecidability to Consistency and Incompleteness

Fact (Consistency)

If \( p \preceq A^\vdash \) and there is \( x \) with \( \neg p \times \) then \( A \not\vdash \bot \).

Proof.

Let \( f \) witness \( p \preceq A^\vdash \). Then \( A \not\vdash f \times \) by \( \neg p \times \) and thus \( A \not\vdash \bot \). \( \square \)
Connection of Undecidability to Consistency and Incompleteness

Fact (Consistency)

If \( p \preceq A \vdash \) and there is \( x \) with \( \neg p \times \) then \( A \not\vdash \bot \).

Proof.

Let \( f \) witness \( p \preceq A \vdash \). Then \( A \not\vdash f \times \) by \( \neg p \times \) and thus \( A \not\vdash \bot \).

Fact (Synthetic Incompleteness)

If \( A \) is complete (\( \forall \varphi. A \vdash \varphi \lor A \vdash \neg \varphi \)) and consistent, then \( A \vdash \) is decidable.
Connection of Undecidability to Consistency and Incompleteness

**Fact (Consistency)**

If $p \leq A^\vdash$ and there is $x$ with $\neg p x$ then $A \not\vdash \bot$.

**Proof.**

Let $f$ witness $p \leq A^\vdash$. Then $A \not\vdash f x$ by $\neg p x$ and thus $A \not\vdash \bot$.

**Fact (Synthetic Incompleteness)**

If $A$ is complete ($\forall \phi. A \vdash \phi \lor A \vdash \neg \phi$) and consistent, then $A^\vdash$ is decidable.

**Proof.**

$A^\vdash$ is enumerable and, given completeness and consistency, also co-enumerable as then $A \not\vdash \phi$ iff $A \vdash \neg \phi$. 
Connection of Undecidability to Consistency and Incompleteness

Fact (Consistency)

If \( p \preceq A^\vdash \) and there is \( x \) with \( \neg p \times \) then \( A \nvdash \bot \).

Proof.

Let \( f \) witness \( p \preceq A^\vdash \). Then \( A \nvdash f \times \) by \( \neg p \times \) and thus \( A \nvdash \bot \).

Fact (Synthetic Incompleteness)

If \( A \) is complete (\( \forall \varphi. A \vdash \varphi \vee A \vdash \neg \varphi \)) and consistent, then \( A^\vdash \) is decidable.

Proof.

\( A^\vdash \) is enumerable and, given completeness and consistency, also co-enumerable as then \( A \nvdash \varphi \) iff \( A \vdash \neg \varphi \). Classically, this is enough to deduce decidability, in our case we need to first observe that \( A^\vdash \) is definite, i.e. that \( A \vdash \varphi \vee A \nvdash \varphi \).
Sketch for Peano Arithmetic

Use axiomatisation PA over standard signature \((0, S, +, \cdot; \equiv)\).
Sketch for Peano Arithmetic

Use axiomatisation PA over standard signature \((0, S, +, \cdot; \equiv)\).

**Diophantine constraints** (cf. Larchey-Wendling and Forster (2019)):

- Instances are lists \(L\) of constraints \(x_i = 1 \mid x_i + x_j = x_k \mid x_i \cdot x_j = x_k\)
- \(L\) is solvable if there is an evaluation \(\eta : \mathbb{N} \to \mathbb{N}\) solving all constraints
Sketch for Peano Arithmetic

Use axiomatisation PA over standard signature \((0, S, +, \cdot; \equiv)\).

**Diophantine constraints** (cf. Larchey-Wendling and Forster (2019)):

- Instances are lists \(L\) of constraints \(x_i = 1 \mid x_i + x_j = x_k \mid x_i \cdot x_j = x_k\)
- \(L\) is solvable if there is an evaluation \(\eta : \mathbb{N} \to \mathbb{N}\) solving all constraints

**Theorem**

\([c_1, \ldots, c_k]\) with maximal index \(x_n\) is solvable iff \(\text{PA} \models \exists n c_1 \land \cdots \land c_k\).
Sketch for Peano Arithmetic

Use axiomatisation PA over standard signature \((0, S, +, \cdot; \equiv)\).

Diophantine constraints (cf. Larchey-Wendling and Forster (2019)):

- Instances are lists \(L\) of constraints \(x_i = 1 \mid x_i + x_j = x_k \mid x_i \cdot x_j = x_k\)
- \(L\) is solvable if there is an evaluation \(\eta : \mathbb{N} \rightarrow \mathbb{N}\) solving all constraints

### Theorem

\[ L = [c_1, \ldots, c_k] \text{ with maximal index } x_n \text{ is solvable iff } \text{PA} \models \exists^n c_1 \land \cdots \land c_k. \]

### Proof.

If \(L\) has solution \(\eta\) instantiate the existential quantifiers with numerals \(\overline{\eta_1}, \ldots, \overline{\eta_n}\). Then the axioms of PA entail the constraints.
Sketch for Peano Arithmetic

Use axiomatisation PA over standard signature \((0, S, +, \cdot; \equiv)\).

**Diophantine constraints** (cf. Larchey-Wendling and Forster (2019)):
- Instances are lists \(L\) of constraints \(x_i = 1\mid x_i + x_j = x_k\mid x_i \cdot x_j = x_k\)
- \(L\) is solvable if there is an evaluation \(\eta : \mathbb{N} \rightarrow \mathbb{N}\) solving all constraints

**Theorem**

\[ L = [c_1, \ldots, c_k] \text{ with maximal index } x_n \text{ is solvable iff } PA \models \exists^n c_1 \land \cdots \land c_k. \]

**Proof.**

If \(L\) has solution \(\eta\) instantiate the existential quantifiers with numerals \(\overline{\eta_1}, \ldots, \overline{\eta_n}\). Then the axioms of PA entail the constraints.
If \(PA \models \exists^n c_1 \land \cdots \land c_k\) use the standard model \(\mathbb{N}\) to extract solution \(\eta\). \(\square\)
Sketch for Peano Arithmetic

Use axiomatisation PA over standard signature \((0, S, +, \cdot; \equiv)\).

Diophantine constraints \(\text{(cf. Larchey-Wendling and Forster (2019))}\\):

- Instances are lists \(L\) of constraints \(x_i = 1 \mid x_i + x_j = x_k \mid x_i \cdot x_j = x_k\)
- \(L\) is solvable if there is an evaluation \(\eta : \mathbb{N} \rightarrow \mathbb{N}\) solving all constraints

**Theorem**

\[L = [c_1, \ldots, c_k] \text{ with maximal index } x_n \text{ is solvable iff } PA \vdash \exists^n c_1 \land \cdots \land c_k.\]

**Proof.**

If \(L\) has solution \(\eta\) instantiate the existential quantifiers with numerals \(\eta_1, \ldots, \eta_n\). Then the axioms of PA entail the constraints.

If \(PA \vdash \exists^n c_1 \land \cdots \land c_k\) use the standard model \(\mathbb{N}\) to extract solution \(\eta\). □

**Fact**

\[L = [c_1, \ldots, c_k] \text{ with maximal index } x_n \text{ is solvable iff } PA \vdash \exists^n c_1 \land \cdots \land c_k.\]
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $\mathcal{T}$ of well-founded trees with constructor $\tau : \forall X. (X \to \mathcal{T}) \to \mathcal{T}$

Equality of trees $s$, $t$ given by isomorphism $s \approx t$.

Membership defined by $s \in \tau X f := \exists x. s \approx f x$.

Set operations implemented by tree operations:

- $\emptyset := \tau \bot$ elim $\bot$.
- $\{ s, t \} := \tau \mathcal{B}(\lambda b. \text{if } b \text{ then } s \text{ else } t)$.
- $\omega := \tau \mathcal{N}(\lambda n. n)$ where $0 := \emptyset$ and $S n := n \cup \{ n \}$.

Axioms needed in Coq:

- EM to really interpret ZF instead of IZF.
- Replacement needs a type-theoretical choice axiom (Werner (1997)).
- Strong quotient axiom for $(\mathcal{T}, \approx)$ suffices (Kirst and Smolka (2019)).

This yields a well-behaved model $S$; quotiented, standard numbers.
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $T$ of well-founded trees with constructor $\tau : \forall X. (X \rightarrow T) \rightarrow T$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $T$ of well-founded trees with constructor $\tau : \forall X. (X \to T) \to T$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $T$ of well-founded trees with constructor $\tau : \forall X. (X \to T) \to T$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \bot \text{elim}_\bot$

Axioms needed in Coq:
- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
- Strong quotient axiom for $(T, \approx)$ suffices (Kirst and Smolka (2019))

This yields a well-behaved model $S$: quotiented, standard numbers
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $T$ of well-founded trees with constructor $\tau : \forall X. (X \to T) \to T$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \bot \text{elim}_\bot$
  - $\{ s, t \} := \tau \big( \lambda b. \text{if } b \text{ then } s \text{ else } t \big)$

Axioms needed in Coq:
- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
- Strong quotient axiom for $(T, \approx)$ suffices (Kirst and Smolka (2019))

This yields a well-behaved model $S$: quotiented, standard numbers
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $\mathcal{T}$ of well-founded trees with constructor $\tau : \forall X. (X \to \mathcal{T}) \to \mathcal{T}$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \bot \text{elim}_\bot$
  - $\{s, t\} := \tau \mathbb{B} (\lambda b. \text{if } b \text{ then } s \text{ else } t)$
  - $\omega := \tau \mathbb{N} (\lambda n. \bar{n})$ where $\bar{0} := \emptyset$ and $\overline{Sn} := \bar{n} \cup \{\bar{n}\}$

Axioms needed in Coq:
- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
- Strong quotient axiom for $(\mathcal{T}, \approx)$ suffices (Kirst and Smolka (2019))

This yields a well-behaved model $S$: quotiented, standard numbers
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $\mathcal{T}$ of well-founded trees with constructor $\tau : \forall X. (X \to \mathcal{T}) \to \mathcal{T}$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \bot \text{ elim}_\bot$
  - $\{ s, t \} := \tau \mathbb{B} (\lambda b. \text{ if } b \text{ then } s \text{ else } t)$
  - $\omega := \tau \mathbb{N} (\lambda n. \bar{n})$ where $\bar{0} := \emptyset$ and $\bar{S}n := \bar{n} \cup \{\bar{n}\}$
  - ...

Axioms needed in Coq:
- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
- Strong quotient axiom for $(\mathcal{T}, \approx)$ suffices (Kirst and Smolka (2019))

This yields a well-behaved model $\mathcal{S}$: quotiented, standard numbers
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $\mathcal{T}$ of well-founded trees with constructor $\tau : \forall X. (X \to \mathcal{T}) \to \mathcal{T}$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \perp \text{elim}_\perp$
  - $\{s, t\} := \tau B (\lambda b. \text{if } b \text{ then } s \text{ else } t)$
  - $\omega := \tau N (\lambda n. \overline{n})$ where $\overline{0} := \emptyset$ and $\overline{Sn} := \overline{n} \cup \{\overline{n}\}$
  - ...

Axioms needed in Coq:

- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $T$ of well-founded trees with constructor $\tau : \forall X. (X \to T) \to T$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \bot \text{elim}_\bot$
  - $\{ s, t \} := \tau \mathbb{B} (\lambda b. \text{if } b \text{ then } s \text{ else } t)$
  - $\omega := \tau \mathbb{N} (\lambda n. \bar{n})$ where $\bar{0} := \emptyset$ and $\bar{S} \bar{n} := \bar{n} \cup \{ \bar{n} \}$
  - ...

Axioms needed in Coq:

- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
- Strong quotient axiom for $(T, \approx)$ suffices (Kirst and Smolka (2019))
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $T$ of well-founded trees with constructor $\tau : \forall X. (X \to T) \to T$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \bot \text{elim}_\bot$
  - $\{s, t\} := \tau \mathbb{B} (\lambda b. \text{if } b \text{ then } s \text{ else } t)$
  - $\omega := \tau \mathbb{N} (\lambda n. \overline{n})$ where $\overline{0} := \emptyset$ and $\overline{S n} := \overline{n} \cup \{\overline{n}\}$
  - ...

Axioms needed in Coq:

- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
- Strong quotient axiom for $(T, \approx)$ suffices (Kirst and Smolka (2019))
- This yields a well-behaved model $S$: quotiented, standard numbers
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \((\emptyset, \{\_\,\_\}, \cup, \mathcal{P}, \omega; \equiv, \in)\).
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \((\emptyset, \{\_\}, \cup, \mathcal{P}, \omega; \equiv, \in)\).

Reduction from PCP:

- **Boolean encoding**: \(\overline{tt} = \{\emptyset\}\) and \(\overline{ff} = \emptyset\)
- **String encoding**: \(ttff\) \(ff\) \(tt = (tt, (ff, (tt, (ff, \emptyset))))\)
- **Stack encoding**: \(\overline{S} = \{(s_1, t_1), \ldots, (s_k, t_k)\}\)
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \( (\emptyset, \{\_, \_\}, \cup, \mathcal{P}, \omega; \equiv, \in) \).

Reduction from PCP:

- Boolean encoding: \( \overline{tt} = \{\emptyset\} \) and \( \overline{ff} = \emptyset \)
- String encoding: \( \overline{ttffftt} = (\overline{tt}, (\overline{ff}, (\overline{tt}, (\overline{ff}, \emptyset)))) \)
- Stack encoding: \( \overline{S} = \{(s_1, t_1), \ldots, (s_k, t_k)\} \)
- Combination encoding: \( S + B := \bigcup_{s/t \in S} \{ (\overline{s}\,x, \overline{t}\,y) \mid (x, y) \in B \} \)
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \((\emptyset, \{-, \_\}, \cup, \mathcal{P}, \omega; \equiv, \in)\).

Reduction from PCP:

- Boolean encoding: \(\overline{\text{tt}} = \{\emptyset\}\) and \(\overline{\text{ff}} = \emptyset\)
- String encoding: \(\overline{\text{ttffft}} = (\overline{\text{tt}}, (\overline{\text{ff}}, (\overline{\text{tt}}, (\overline{\text{ff}}, \emptyset))))\)
- Stack encoding: \(\overline{S} = \{(s_1, t_1), \ldots, (s_k, t_k)\}\)
- Combination encoding: \(S \leftrightarrow B := \bigcup_{s/t \in S} \{(\overline{s}x, \overline{t}y) \mid (x, y) \in B\}\)
- \(f \triangleright n := (\emptyset, \overline{S}) \in f \land \forall (k, B) \in f. k \in n \rightarrow (k + 1, S \leftrightarrow B) \in f\)
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \((\emptyset, \{\_\_\}, \cup, \mathcal{P}, \omega; \equiv, \in)\).

Reduction from PCP:

- Boolean encoding: \(\overline{\text{tt}} = \{\emptyset\}\) and \(\overline{\text{ff}} = \emptyset\)
- String encoding: \(\text{ttfffftt} = (\overline{\text{tt}}, (\overline{\text{ff}}, (\overline{\text{tt}}, (\overline{\text{ff}}, \emptyset))))\)
- Stack encoding: \(S = \{(\overline{s_1}, \overline{t_1}), \ldots, (\overline{s_k}, \overline{t_k})\}\)
- Combination encoding: \(S \leftrightarrow B := \bigcup_{s/t \in S}\{(\overline{s}x, \overline{t}y) \mid (x, y) \in B\}\)
- \(f \triangleright n := (\emptyset, \overline{S}) \in f \land \forall (k, B) \in f. k \in n \rightarrow (k + 1, S \leftrightarrow B) \in f\)

\[
\varphi_S := \exists f, n, B, x. n \in \omega \land f \triangleright n \land (n, B) \in f \land (x, x) \in B
\]
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \((\emptyset, \{\_ \}, \cup, \mathcal{P}, \omega; \equiv, \in)\).

Reduction from PCP:

- Boolean encoding: \(\overline{\text{tt}} = \{\emptyset\}\) and \(\overline{\text{ff}} = \emptyset\)
- String encoding: \(\overline{\text{tt ff ff tt}} = (\overline{\text{tt}}, (\overline{\text{ff}}, (\overline{\text{tt}}, (\overline{\text{ff}}, \emptyset))))\)
- Stack encoding: \(\overline{S} = \{(s_1, t_1), \ldots, (s_k, t_k)\}\)
- Combination encoding: \(S \oplus B := \bigcup_{s/t \in S} \{(\overline{s}, \overline{t}y) \mid (x, y) \in B\}\)
- \(f \triangleright n := (\emptyset, \overline{S}) \in f \land \forall(k, B) \in f. k \in n \rightarrow (k + 1, S \oplus B) \in f\)

\[
\varphi_S := \exists f, n, B, x. n \in \omega \land f \triangleright n \land (n, B) \in f \land (x, x) \in B
\]

Theorem

PCP \(S\) iff ZF \(\models \varphi_S\) and PCP \(S\) iff ZF \(\vdash \varphi_S\).
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \((\emptyset, \{\_, \_\}, \cup, \mathcal{P}, \omega; \equiv, \in)\).

Reduction from PCP:

- Boolean encoding: \(\texttt{tt} = \{\emptyset\}\) and \(\texttt{ff} = \emptyset\)
- String encoding: \(\texttt{tt ff ff tt} = (\texttt{tt}, (\texttt{ff}, (\texttt{tt}, (\texttt{ff}, \emptyset))))\)
- Stack encoding: \(\mathcal{S} = \{(s_1, t_1), \ldots, (s_k, t_k)\}\)
- Combination encoding: \(\mathcal{S} \oplus \mathcal{B} := \bigcup_{s/t \in \mathcal{S}} \{(sx, ty) | (x, y) \in \mathcal{B}\}\)
- Stack encoding: \(f \triangleright n := (\emptyset, \mathcal{S}) \in f \land \forall (k, B) \in f. k \in n \rightarrow (k + 1, \mathcal{S} \oplus \mathcal{B}) \in f\)

\[\varphi_S := \exists f, n, B, x. n \in \omega \land f \triangleright n \land (n, B) \in f \land (x, x) \in B\]

Theorem

PCP \(S\) iff \(ZF \models \varphi_S\) and PCP \(S\) iff \(ZF \vdash \varphi_S\).

Proof.

Direction \(\rightarrow\) by proofs in ZF and \(\leftarrow\) relies on standard model \(S\). □
Conclusion
Ongoing and Future Work

- Undecidability and incompleteness of finitary set theories
- Minimalistic undecidability proof for the binary signature
- Undecidability and incompleteness of second-order logic
- Constructive analysis of Tennenbaum’s theorem
- Stronger incompleteness results (only using consistency, explicit Gödel sentence)
- Constructive completeness of intuitionistic epistemic logic
- Engineering: tool support, connect Coq developments
Take-Home Messages

- Metamathematics: rewarding to revisit in formal setting
- Mechanisation: feasible with right setup and suitable proof strategies
- Synthetic computability: elegant formalism, shortcut to algorithmic results
- Constructive type theory: ideal framework for (constructive) reverse mathematics
Take-Home Messages

- Metamathematics: rewarding to revisit in formal setting
- Mechanisation: feasible with right setup and suitable proof strategies
- Synthetic computability: elegant formalism, shortcut to algorithmic results
- Constructive type theory: ideal framework for (constructive) reverse mathematics

Thank You!
Bibliography I


Bibliography IV


The Coq Development Team (2021). The coq proof assistant.


