Analysing First-Order Logic in Constructive Type Theory

Dominik Kirst   Dominik Wehr   Yannick Forster

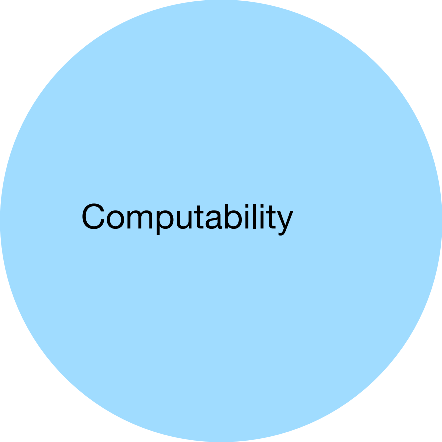
ANU Logic Seminar
December 16th/17th, 2021

SAARLAND UNIVERSITY
COMPUTER SCIENCE
UNIVERSITY OF GOTHENBURG
Inria

What are we working on?*

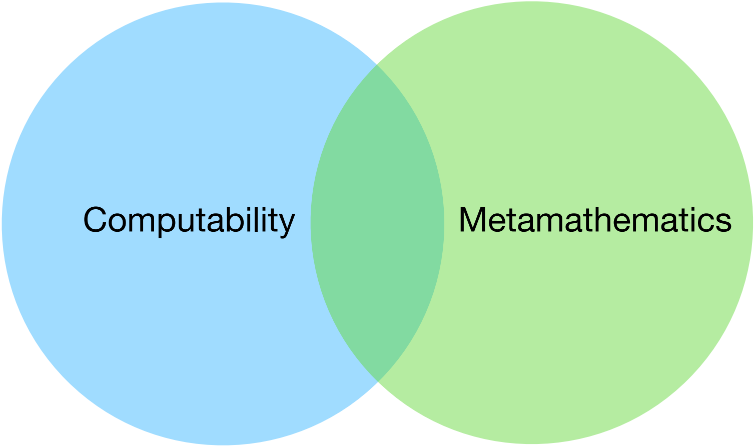
*With members, students and collaborators of the Programming Systems Lab at Saarland University

What are we working on?*



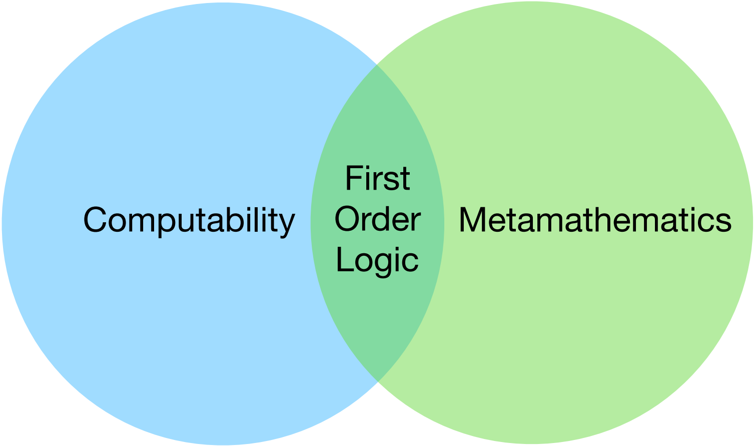
*With members, students and collaborators of the Programming Systems Lab at Saarland University

What are we working on?*



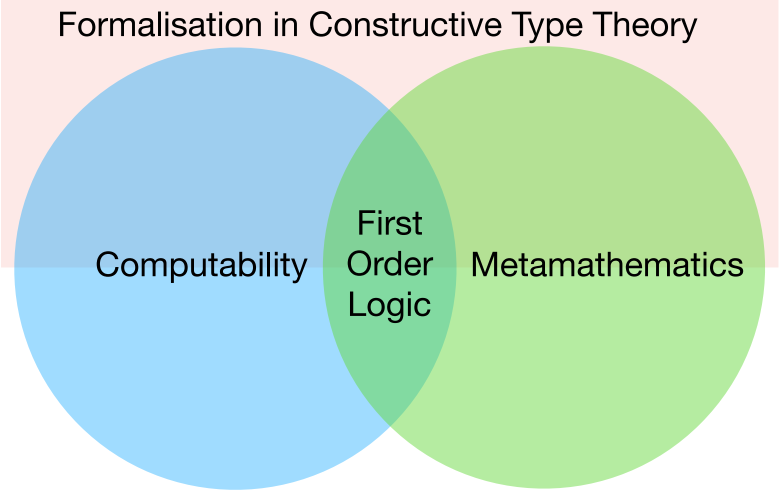
*With members, students and collaborators of the Programming Systems Lab at Saarland University

What are we working on?*



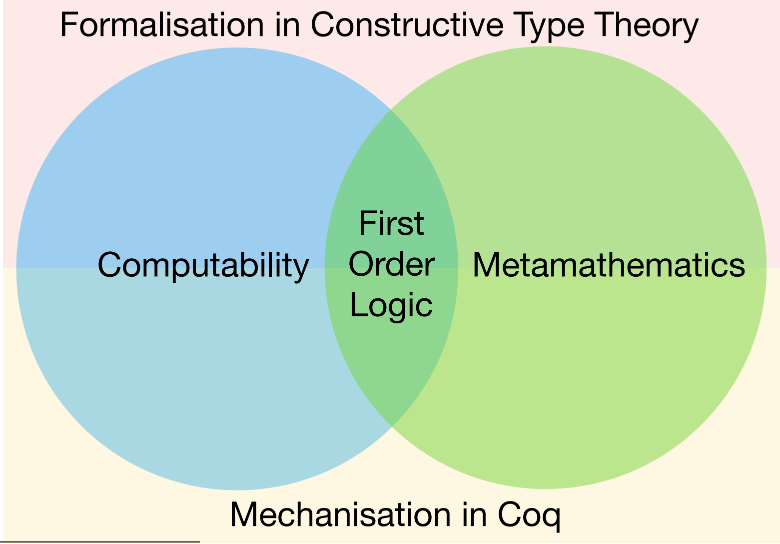
*With members, students and collaborators of the Programming Systems Lab at Saarland University

What are we working on?*



*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

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)

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

First-Order Logic in Coq

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

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

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

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

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

- Inductive types: $\mathbb{0}$, $\mathbb{1}$, $\mathbb{B}$, $\mathbb{N}$, 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: $\bot$, $\top$, $\to$, $\wedge$, $\vee$, $\forall$, $\exists$

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

- Inductive types: $\mathbb{0}$, $\mathbb{1}$, $\mathbb{B}$, $\mathbb{N}$, 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:

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

- Inductive types: $\mathbb{0}$, $\mathbb{1}$, $\mathbb{B}$, $\mathbb{N}$, 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

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

- Inductive types: $\mathbb{0}$, $\mathbb{1}$, $\mathbb{B}$, $\mathbb{N}$, 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: $\bot$, $\top$, $\to$, $\wedge$, $\vee$, $\forall$, $\exists$

Specifics of the induced logic:

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

Calculus of Inductive Constructions (CIC)

Features as implemented in Coq's type theory:

- Inductive types: $\mathbb{0}$, $\mathbb{1}$, $\mathbb{B}$, $\mathbb{N}$, 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: $\mathbb{0}$, $\mathbb{1}$, $\mathbb{B}$, $\mathbb{N}$, 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
- Proof-irrelevant: no computational content extractable from $\lor$ and $\exists$

Representing Syntax

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

$$t \;:\; \mathsf{Term} \;::=\; x \mid f\,\vec{t} \qquad\qquad (x : \mathbb{N},\; f : \mathcal{F}_\Sigma,\; \vec{t} : \mathsf{Term}^{|f|})$$

$$\varphi, \psi \;:\; \mathsf{Form} \;::=\; \dot{\bot} \mid P\,\vec{t} \mid \varphi \,\dot{\square}\, \psi \mid \dot{\nabla} \varphi \qquad\qquad (P : \mathcal{P}_\Sigma,\; \vec{t} : \mathsf{Term}^{|P|})$$

Representing Syntax

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

$$t \;:\; \mathsf{Term} \;::=\; x \mid f\,\vec{t} \qquad\qquad (x : \mathbb{N},\; f : \mathcal{F}_\Sigma,\; \vec{t} : \mathsf{Term}^{|f|})$$

$$\varphi, \psi \;:\; \mathsf{Form} \;::=\; \dot{\bot} \mid P\,\vec{t} \mid \varphi \,\dot{\square}\, \psi \mid \dot{\nabla} \varphi \qquad\qquad (P : \mathcal{P}_\Sigma,\; \vec{t} : \mathsf{Term}^{|P|})$$

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

Representing Syntax

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

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

$$\varphi, \psi : \text{Form} ::= \dot{\bot} \mid P\,\vec{t} \mid \varphi \,\dot{\square}\, \psi \mid \dot{\nabla}\varphi \qquad\qquad (P : \mathcal{P}_\Sigma,\ \vec{t} : \text{Term}^{|P|})$$

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

Coq implementation:

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

Representing Semantics

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

$$\_^{\mathcal{M}} \; : \; \forall f : \mathcal{F}_\Sigma.\, D^{|f|} \to D \qquad\qquad \_^{\mathcal{M}} \; : \; \forall P : \mathcal{P}_\Sigma.\, D^{|P|} \to \mathbb{P}$$

Representing Semantics

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

$$\_^{\mathcal{M}} \;:\; \forall f : \mathcal{F}_\Sigma.\, D^{|f|} \to D \qquad\qquad \_^{\mathcal{M}} \;:\; \forall P : \mathcal{P}_\Sigma.\, D^{|P|} \to \mathbb{P}$$

For assignments $\rho : \mathbb{N} \to D$ define evaluation $\hat{\rho}\, t$ and satisfaction $\mathcal{M} \vDash_\rho \varphi$:

$$\hat{\rho}\, x \;:=\; \rho\, x \qquad\qquad \hat{\rho}\,(f\, \vec{t}\,) \;:=\; f^{\mathcal{M}}\,(\hat{\rho}\, \vec{t}\,)$$

$$\mathcal{M} \vDash_\rho P\, \vec{t} \;:=\; P^{\mathcal{M}}\,(\hat{\rho}\, \vec{t}\,) \qquad\qquad \mathcal{M} \vDash_\rho \varphi \,\dot{\Box}\, \psi \;:=\; \mathcal{M} \vDash_\rho \varphi \,\Box\, \mathcal{M} \vDash_\rho \psi$$

$$\mathcal{M} \vDash_\rho \dot{\bot} \;:=\; \bot \qquad\qquad \mathcal{M} \vDash_\rho \dot{\nabla} \varphi \;:=\; \nabla a : D.\, \mathcal{M} \vDash_{a \cdot \rho} \varphi$$

Obtain derived notation for theories $\mathcal{T} : \mathsf{Form} \to \mathbb{P}$ like $\mathcal{M} \vDash \mathcal{T}$, and $\mathcal{T} \vDash \varphi$.

Representing Semantics

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

$$\_^{\mathcal{M}} \;:\; \forall f : \mathcal{F}_\Sigma.\, D^{|f|} \to D \qquad\qquad \_^{\mathcal{M}} \;:\; \forall P : \mathcal{P}_\Sigma.\, D^{|P|} \to \mathbb{P}$$

For assignments $\rho : \mathbb{N} \to D$ define evaluation $\hat{\rho}\, t$ and satisfaction $\mathcal{M} \vDash_\rho \varphi$:

$$\hat{\rho}\, x \;:=\; \rho\, x \qquad\qquad \hat{\rho}\,(f\,\vec{t}\,) \;:=\; f^{\mathcal{M}}\,(\hat{\rho}\,\vec{t}\,)$$

$$\mathcal{M} \vDash_\rho P\,\vec{t} \;:=\; P^{\mathcal{M}}\,(\hat{\rho}\,\vec{t}\,) \qquad\qquad \mathcal{M} \vDash_\rho \varphi \,\dot{\Box}\, \psi \;:=\; \mathcal{M} \vDash_\rho \varphi \,\Box\, \mathcal{M} \vDash_\rho \psi$$

$$\mathcal{M} \vDash_\rho \dot{\bot} \;:=\; \bot \qquad\qquad \mathcal{M} \vDash_\rho \dot{\nabla} \varphi \;:=\; \nabla a : D.\, \mathcal{M} \vDash_{a \cdot \rho} \varphi$$

Obtain derived notation for theories $\mathcal{T} : \mathsf{Form} \to \mathbb{P}$ like $\mathcal{M} \vDash \mathcal{T}$, and $\mathcal{T} \vDash \varphi$.

Coq implementation: type class for models, structurally recursive functions

Representing Deduction

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

$$\dots$$

$$\frac{\Gamma \vdash \dot{\forall}\varphi}{\Gamma \vdash \varphi[t]}\ AE \qquad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \dot{\forall}\varphi}\ AI$$

$$\dots$$

Representing Deduction

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

$$\cdots$$

$$\frac{\Gamma \vdash \dot\forall \varphi}{\Gamma \vdash \varphi[t]} \; AE \qquad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \dot\forall \varphi} \; AI$$

$$\cdots$$

- Given Γ and φ 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 $\mathcal{L}(\text{Form}) \to \text{Form} \to \mathbb{P}$:

$$\ldots$$

$$\frac{\Gamma \vdash \dot\forall \varphi}{\Gamma \vdash \varphi[t]}\ AE \qquad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \dot\forall \varphi}\ AI$$

$$\ldots$$

- Given Γ and φ 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 $\Gamma \vdash_c ((\varphi \dot\to \psi) \dot\to \varphi) \dot\to \varphi$

Representing Deduction

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

$$\dots$$

$$\frac{\Gamma \vdash \dot\forall \varphi}{\Gamma \vdash \varphi[t]}\ AE \qquad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \dot\forall \varphi}\ AI$$

$$\dots$$

- Given Γ and φ 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 $\Gamma \vdash_c ((\varphi \dot\to \psi) \dot\to \varphi) \dot\to \varphi$
- Natural generalisation to $\mathcal{T} \vdash \varphi$ by $\exists \Gamma \subseteq \mathcal{T}.\, \Gamma \vdash \varphi$

Representing Deduction

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

$$\ldots$$

$$\frac{\Gamma \vdash \dot\forall \varphi}{\Gamma \vdash \varphi[t]}\ AE \qquad \frac{\Gamma[+1] \vdash \varphi}{\Gamma \vdash \dot\forall \varphi}\ AI$$

$$\ldots$$

- Given Γ and φ 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 $\Gamma \vdash_c ((\varphi \dot\to \psi) \dot\to \varphi) \dot\to \varphi$
- 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

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* $\vdash_c$.

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 $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$ *one can construct functions*

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

Curious Observations

Fact

The intuitionistic deduction system is sound for Tarski semantics, i.e. $\Gamma \vdash_i \varphi$ *implies* $\Gamma \vDash \varphi$.
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 $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$ *one can construct functions*

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

Synthetic computability: equality on formulas is decidable ($\mathcal{D}$) and provability is enumerable ($\mathcal{E}$)

Completeness*

*F., K., and W. at LFCS'20.

Completeness: Syntax fragments

A lot of our work is restricted to the fragment

$$t : \mathsf{Term} ::= x \mid f\,\overline{t} \qquad\qquad n : \mathbb{N}, f : \Sigma$$

$$\varphi : \mathsf{Form}^* ::= \dot{\bot} \mid P\,\overline{t} \mid \varphi \dot{\to} \psi \mid \dot{\forall}\varphi \qquad\qquad P : \Sigma$$

Completeness: Syntax fragments

A lot of our work is restricted to the fragment

$$t : \mathsf{Term} ::= x \mid f\,\overline{t} \qquad\qquad n : \mathbb{N}, f : \Sigma$$

$$\varphi : \mathsf{Form}^* ::= \dot{\bot} \mid P\,\overline{t} \mid \varphi \dot{\to} \psi \mid \dot{\forall}\varphi \qquad\qquad P : \Sigma$$

Definition

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

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

Interpreting

$$\mathcal{M} \vDash_\rho \dot{\bot} :\Leftrightarrow \bot^{\mathcal{M}}$$

Tarski models: Classes of models

Definition

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

$$\mathcal{M} \vDash_\rho \dot\neg\dot\neg\varphi \,\dot\rightarrow\, \varphi$$

Tarski models: Classes of models

Definition

A model $\mathcal{M}$ is called classical* if for all ρ and φ : Form*

$$\mathcal{M} \vDash_\rho \dot\neg\dot\neg\varphi \mathrel{\dot\to} \varphi$$

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

Tarski models: Classes of models

Definition

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

$$\mathcal{M} \vDash_\rho \dot\neg\dot\neg\varphi \mathbin{\dot\to} \varphi$$

A model $\mathcal{M}$ is called standard if $\perp^{\mathcal{M}}$ is contradictory.
A model $\mathcal{M}$ is called exploding* (due to Veldman (1976)) if for all ρ and formulas $\varphi : \mathrm{Form}^*$

$$\mathcal{M} \vDash_\rho \dot\perp \mathbin{\dot\to} \varphi$$

Tarski models: Classes of models

Definition

A model $\mathcal{M}$ is called classical* if for all ρ and φ : Form*

$$\mathcal{M} \vDash_\rho \dot{\neg}\dot{\neg}\varphi \dot{\to} \varphi$$

A model $\mathcal{M}$ is called standard if $\perp^{\mathcal{M}}$ is contradictory.
A model $\mathcal{M}$ is called exploding* (due to Veldman (1976)) if for all ρ and formulas φ : Form*

$$\mathcal{M} \vDash_\rho \dot{\perp} \dot{\to} \varphi$$

Fact

Every standard model is exploding. The converse need not hold.*

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 $\mathcal{M} \vDash_\rho \varphi$*
- *If $\mathcal{M} \vDash_\rho \dot{\bot}$ then $\mathcal{T} \vdash \dot{\bot}$*

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 $\mathcal{M} \vDash_\rho \varphi$*
- *If $\mathcal{M} \vDash_\rho \dot\bot$ then $\mathcal{T} \vdash \dot\bot$*

Proof.

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

$\square$

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 $\mathcal{M} \vDash_\rho \varphi$*
- *If $\mathcal{M} \vDash_\rho \dot\bot$ then $\mathcal{T} \vdash \dot\bot$*

Proof.

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

1. Henkin axioms: Add all formulas $\varphi_n(n) \dot\to \dot\forall \varphi_n$

□

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 $\mathcal{M} \vDash_\rho \varphi$*
- *If $\mathcal{M} \vDash_\rho \dot{\bot}$ then $\mathcal{T} \vdash \dot{\bot}$*

Proof.

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

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

□

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 $\mathcal{M} \vDash_\rho \varphi$*
- *If $\mathcal{M} \vDash_\rho \dot\perp$ then $\mathcal{T} \vdash \dot\perp$*

Proof.

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

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

The term-model induced by Ω fulfills all desiderata. □

Tarski completeness: The standard case

Theorem

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

Tarski completeness: The standard case

Theorem

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

Proof.

$\square$

Tarski completeness: The standard case

Theorem

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

Proof.

Suppose $\mathcal{T} \nvdash \varphi$, meaning $\mathcal{T}' := \mathcal{T} \cup \{\dot{\neg}\varphi\}$ is consistent.

□

Tarski completeness: The standard case

Theorem

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

Proof.

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

1. If $\mathcal{T} \cup \{\dot{\neg}\varphi\} \vdash \varphi$ then $\mathcal{M} \vDash_\rho \varphi$
2. If $\mathcal{M} \vDash_\rho \dot{\perp}$ then $\mathcal{T} \cup \{\dot{\neg}\varphi\} \vdash \dot{\perp}$

$\square$

Tarski completeness: The standard case

Theorem

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

Proof.

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

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

Then

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

□

Tarski completeness: The standard case

Theorem

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

Proof.

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

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

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 $\mathcal{M} \vDash_\rho \varphi$.

□

Tarski completeness: The standard case

Theorem

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

Proof.

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

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

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 $\mathcal{M} \vDash_\rho \dot{\neg}\varphi$.

$\square$

Tarski completeness: The standard case

Theorem

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

Fact

$$\mathcal{T} \vDash \varphi \to \mathcal{T} \vdash \varphi \quad \textit{iff} \quad \neg\neg\mathcal{T} \vdash \varphi \to \mathcal{T} \vdash \varphi.$$

Tarski completeness: The standard case

Theorem

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

Fact

$$\mathcal{T} \vDash \varphi \to \mathcal{T} \vdash \varphi \quad \textit{iff} \quad \neg\neg\mathcal{T} \vdash \varphi \to \mathcal{T} \vdash \varphi.$$

Corollary

$\forall \mathcal{T}, \varphi.\ \mathcal{T} \vDash \varphi \to \mathcal{T} \vdash \varphi$ is not provable in the CIC.

Tarski completeness: Examining the standard case*

Theorem

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

Proof.

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

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

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 $\mathcal{M} \vDash_\rho \dot\neg\varphi$.

□

*Following Herbelin and Ilik (2016)

Tarski completeness: Examining the standard case*

Theorem

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

Proof.

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

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

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 $\mathcal{M} \vDash_\rho \dot\neg\varphi$.

□

*Following Herbelin and Ilik (2016)

Tarski completeness: Examining the standard case*

Theorem

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

Proof.

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

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

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 $\mathcal{M} \vDash_\rho \dot{\neg}\varphi$.

□

*Following Herbelin and Ilik (2016)

Tarski completeness: The exploding case

Theorem

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

Proof.

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

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

Then

- By (1) $\mathcal{M}, \rho$ is classical* and $\mathcal{M}, \rho$ **exploding***.
- By (1) we have $\mathcal{M} \vDash_\rho \mathcal{T}$ and thus $\mathcal{M} \vDash_\rho \varphi$.
- By (1) we also have $\mathcal{M} \vDash_\rho \dot\neg\varphi$, **meaning** $\mathcal{M} \vDash_\rho \dot\bot$.
- **By (2) this means $\mathcal{T} \cup \{\dot\neg\varphi\} \vdash \dot\bot$ 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 $f : D^{|f|} \to D$
- 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 $\vec{d} : D^{|P|}$

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 $\rho : \mathcal{V} \to D$ we define

$$w \Vdash_\rho \dot{\bot} :\Leftrightarrow \bot_w$$
$$w \Vdash_\rho P\,\vec{t} :\Leftrightarrow P_w\,\vec{t^\rho}$$
$$w \Vdash_\rho \dot{\forall}\varphi :\Leftrightarrow \forall d : D.\ w \Vdash_{d\cdot\rho} \varphi$$
$$w \Vdash_\rho \varphi \dot{\to} \psi :\Leftrightarrow \forall w \leq v.\ v \Vdash_\rho \varphi \to v \Vdash_\rho \psi$$

Kripke models: Semantics

Definition

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

$$w \Vdash_\rho \dot{\forall}\varphi :\Leftrightarrow \forall d : D.\ w \Vdash_{d \cdot \rho} \varphi$$
$$w \Vdash_\rho \varphi \mathrel{\dot{\to}} \psi :\Leftrightarrow \forall w \leq v.\ v \Vdash_\rho \varphi \to v \Vdash_\rho \psi$$

Kripke models: Semantics

Definition

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

$$w \Vdash_\rho \dot{\bot} :\Leftrightarrow \bot_w$$
$$w \Vdash_\rho P\,\vec{t} :\Leftrightarrow P_w\,\vec{t^\rho}$$
$$w \Vdash_\rho \dot{\forall}\varphi :\Leftrightarrow \forall d : D.\ w \Vdash_{d\cdot\rho} \varphi$$
$$w \Vdash_\rho \varphi \dot{\to} \psi :\Leftrightarrow \forall w \leq v.\ v \Vdash_\rho \varphi \to v \Vdash_\rho \psi$$

Fact

If $w \leq v$ then $w \Vdash_\rho \varphi$ entails $v \Vdash_\rho \varphi$.

Kripke models: Classes of models

Definition

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

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

Kripke models: Classes of models

Definition

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

$$w \Vdash_\rho \dot{\bot} \dot{\to} \varphi \text{ for every } w : \mathcal{W} \text{ and } \varphi.$$

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

Kripke models: Classes of models

Definition

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

$$w \Vdash_\rho \dot{\bot} \mathbin{\dot{\to}} \varphi \text{ for every } w : \mathcal{W} \text{ and } \varphi.$$

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

Definition

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

$$(\forall \psi \in \Gamma.\ w \Vdash_\rho \psi) \to w \Vdash_\rho \varphi$$

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

LJT: Focused Sequents

$$\rightarrow\mathsf{L}\,\frac{\Gamma \Rightarrow \psi \quad \Gamma; \theta \Rightarrow \varphi}{\Gamma; \psi \dot{\rightarrow} \theta \Rightarrow \varphi} \qquad \forall\mathsf{L}\,\frac{\Gamma; \psi[t] \Rightarrow \varphi}{\Gamma; \dot{\forall}\psi \Rightarrow \varphi}$$

$$\rightarrow\mathsf{R}\,\frac{\Gamma, \varphi \Rightarrow \psi}{\Gamma \Rightarrow \varphi \dot{\rightarrow} \psi} \qquad \forall\mathsf{R}\,\frac{\uparrow\Gamma \Rightarrow \varphi}{\Gamma \Rightarrow \dot{\forall}\varphi}$$

$$\mathsf{Ax}\,\frac{}{\Gamma; \varphi \Rightarrow \varphi} \qquad \mathsf{Focus}\,\frac{\Gamma; \psi \Rightarrow \varphi \quad \psi \in \Gamma}{\Gamma \Rightarrow \varphi} \qquad \mathsf{Exp}\,\frac{\uparrow\Gamma \Rightarrow \dot{\bot}}{\Gamma \Rightarrow \varphi}$$

Kripke completeness: The exploding case*

Definition

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

$$P_\Gamma \vec{t} := \Gamma \Rightarrow P\,\vec{t} \qquad \bot_\Gamma := \Gamma \Rightarrow \dot{\bot}$$

*Following Herbelin and Lee (2009)

Kripke completeness: The exploding case*

Definition

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

$$P_\Gamma \vec{t} := \Gamma \Rightarrow P\,\vec{t} \qquad \bot_\Gamma := \Gamma \Rightarrow \dot{\bot}$$

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 $(\mathcal{L}(\mathsf{Form}), \subseteq)$ and the domain Term, taking

$$P_\Gamma \vec{t} := \Gamma \Rightarrow P\,\vec{t} \qquad \perp_\Gamma := \Gamma \Rightarrow \dot{\perp}$$

Fact

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

Lemma

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

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

*Following Herbelin and Lee (2009)

Kripke completeness: The exploding case II

Lemma

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

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

Kripke completeness: The exploding case II

Lemma

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

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

Corollary

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

Kripke completeness: The exploding case II

Lemma

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

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

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 Δ.

□

Kripke completeness: The exploding case II

Lemma

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

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

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 Δ.

- Using (2) we conclude $\Gamma \Vdash_{\mathsf{id}} \Gamma$

$\square$

Kripke completeness: The exploding case II

Lemma

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

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

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 Δ.

- Using (2) we conclude $\Gamma \Vdash_{\mathsf{id}} \Gamma$
- Per assumption thus $\Gamma \Vdash_{\mathsf{id}} \varphi$ and $\Gamma \Rightarrow \varphi$ by (1)

□

Kripke completeness: The standard case

Definition

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

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

Kripke completeness: The standard case

Definition

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

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

Fact

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

Kripke completeness: The standard case

Definition

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

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

Fact

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

Fact

$\Gamma \Vdash^s \varphi \rightarrow \Gamma \Rightarrow \varphi$ *iff* $\neg\neg(\Gamma \vdash \varphi) \rightarrow \Gamma \vdash \varphi$.

Completeness: Other semantics

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

- Heyting algebras wrt. intuitionistic FOL
- Inuitionistic 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
- Intuitionstic 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.

*F., K., and W. at LFCS'20.

Sufficient and necessary axioms for completeness

Is there a well-known axiom A with

$$A \to \forall \Gamma \varphi . \Gamma \vDash \varphi \to \Gamma \vdash \varphi$$

Sufficient and necessary axioms for completeness

Is there a well-known axiom A with

$$A \leftrightarrow \forall \Gamma \varphi. \Gamma \vDash \varphi \rightarrow \Gamma \vdash \varphi$$

Sufficient and necessary axioms for completeness

Is there a well-known axiom A with

$$A \leftrightarrow \forall \Gamma \varphi.\ \neg(\Gamma \nvdash \varphi) \to \Gamma \vdash \varphi$$

Sufficient and necessary axioms for completeness

Are there well-known axioms A and A_2 with

$$A \leftrightarrow \forall \Gamma \varphi.\ \neg(\Gamma \nvdash \varphi) \to \Gamma \vdash \varphi$$

$$A_2 \leftrightarrow \forall \mathcal{T} \varphi.\ \neg(\mathcal{T} \nvdash \varphi) \to \mathcal{T} \vdash \varphi$$

Sufficient and necessary axioms for completeness

Is there a well-known axiom-scheme A with

$$A(P) \leftrightarrow \forall \mathcal{T}.\, P\mathcal{T} \to \neg(\mathcal{T} \nvdash \varphi) \to \mathcal{T} \vdash \varphi$$

Slightly confusing literature review

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?

$$\mathsf{MP} \quad := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is primitive recursive} \to \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

Slightly confusing literature review

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

$$\mathsf{MP} \quad := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is primitive recursive} \to \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

$$\mathsf{MP} \quad := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is} \qquad\qquad \textit{recursive} \to \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

Slightly confusing literature review

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

$$\mathsf{MP} := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is primitive recursive} \to \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

$$\mathsf{MP} := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is} \qquad \textit{recursive} \to \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

$$\mathsf{MP} := \forall f : \mathbb{N} \to \mathbb{B}. \qquad \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

Slightly confusing literature review

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

$$\mathsf{MP} \quad := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is primitive recursive} \to \neg\neg(\exists n.\, fn = \mathsf{true}) \to \exists n.\, fn = \mathsf{true}$$

$$\mathsf{MP} \quad := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is} \qquad\qquad \textit{recursive} \to \neg\neg(\exists n.\, fn = \mathsf{true}) \to \exists n.\, fn = \mathsf{true}$$

$$\mathsf{MP} \quad := \forall f : \mathbb{N} \to \mathbb{B}. \qquad\qquad \neg\neg(\exists n.\, fn = \mathsf{true}) \to \exists n.\, fn = \mathsf{true}$$

$$\mathsf{MP} \quad := \forall X.\, X \textit{ is discrete} \to \forall p : X \to \mathbb{P}.\, \mathcal{E}p \to \forall x.\, \neg\neg px \to px$$

Slightly confusing literature review

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

$$\mathsf{MP}_{\mathsf{pr}} := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is primitive recursive} \to \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

$$\mathsf{MP}_{\mathsf{L}} := \forall f : \mathbb{N} \to \mathbb{B}. f \textit{ is} \qquad \textit{recursive} \to \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

$$\mathsf{MP}_{\mathsf{TT}} := \forall f : \mathbb{N} \to \mathbb{B}. \qquad \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

$$\mathsf{MP}_{\mathcal{E}} := \forall X.\ X \textit{ is discrete} \to \forall p : X \to \mathbb{P}.\ \mathcal{E}p \to \forall x.\ \neg\neg px \to px$$

$$\begin{tikzcd}
\mathsf{LEM} \arrow[d] & \\
\mathsf{MP}_{\mathsf{TT}} \arrow[r, bend left] & \mathsf{MP}_{\mathcal{E}} \arrow[l, bend left] \arrow[d] \\
\mathsf{MP}_{\mathsf{pr}} \arrow[r, bend left] & \mathsf{MP}_{\mathsf{L}} \arrow[l, bend left]
\end{tikzcd}$$

Constructive analysis of the completeness theorem

$$\begin{aligned}
\mathsf{LEM} &\leftrightarrow \forall \mathcal{T} \varphi. & \neg(\mathcal{T} \not\vDash \varphi) \to \Gamma \vdash \varphi \\
\mathsf{MP} &\leftrightarrow \forall \mathcal{T} \varphi.\ \mathcal{E}\mathcal{T} \to & \neg(\mathcal{T} \not\vDash \varphi) \to \mathcal{T} \vdash \varphi \\
\mathsf{MP}_\mathsf{L} &\leftrightarrow \forall \mathcal{T} \varphi.\ \mathcal{E}_\mathsf{L}\mathcal{T} \to & \neg(\mathcal{T} \not\vDash \varphi) \to \mathcal{T} \vdash \varphi \\
&\leftrightarrow \forall \Gamma \varphi. & \neg(\Gamma \not\vDash \varphi) \to \Gamma \vdash \varphi
\end{aligned}$$

Constructive analysis of the completeness theorem

$$\mathsf{DNE}(P : \forall X.\,(X \to \mathbb{P}) \to \mathbb{P}) := \forall Xp.\ PXp \to \forall x.\ \neg\neg px \to px$$

$$\begin{aligned}
\mathsf{LEM} &\leftrightarrow \forall \mathcal{T}\varphi.\ & \neg(\mathcal{T} \nvdash \varphi) \to \Gamma \vdash \varphi \quad & \leftrightarrow \mathsf{DNE}(\lambda Xp.\ \top) \\
\mathsf{MP} &\leftrightarrow \forall \mathcal{T}\varphi.\ \mathcal{E}\mathcal{T} \to & \neg(\mathcal{T} \nvdash \varphi) \to \mathcal{T} \vdash \varphi \quad & \leftrightarrow \mathsf{DNE}(\lambda Xp.\ \mathcal{D}X \wedge \mathcal{E}\ p) \\
\mathsf{MP_L} &\leftrightarrow \forall \mathcal{T}\varphi.\ \mathcal{E}_{\mathsf{L}}\mathcal{T} \to & \neg(\mathcal{T} \nvdash \varphi) \to \mathcal{T} \vdash \varphi \quad & \leftrightarrow \mathsf{DNE}(\lambda Xp.\ \mathcal{D}_{\mathsf{L}}X \wedge \mathcal{E}_{\mathsf{L}}p) \\
&\leftrightarrow \forall \Gamma\varphi.\ & \neg(\Gamma \nvdash \varphi) \to \Gamma \vdash \varphi \quad & \leftrightarrow \mathsf{DNE}(\lambda Xp.\ p = \lambda \mathsf{s}.\exists \mathsf{t}.\mathsf{s} \triangleright \mathsf{t})
\end{aligned}$$

Constructive analysis of the completeness theorem

$$\mathrm{DNE}(P : \forall X.\,(X \to \mathbb{P}) \to \mathbb{P}) := \forall Xp.\ PXp \to \forall x.\ \neg\neg px \to px$$

$$\begin{aligned}
\mathrm{LEM} &\leftrightarrow \forall \mathcal{T}\varphi. && \neg(\mathcal{T} \nvdash \varphi) \to \Gamma \vdash \varphi && \leftrightarrow \mathrm{DNE}(\lambda Xp.\ \top) \\
\mathrm{MP} &\leftrightarrow \forall \mathcal{T}\varphi.\ \mathcal{E}\mathcal{T} && \to \neg(\mathcal{T} \nvdash \varphi) \to \mathcal{T} \vdash \varphi && \leftrightarrow \mathrm{DNE}(\lambda Xp.\ \mathcal{D}X \ \wedge \mathcal{E}\ p) \\
\mathrm{MP_L} &\leftrightarrow \forall \mathcal{T}\varphi.\ \mathcal{E}_\mathsf{L}\mathcal{T} && \to \neg(\mathcal{T} \nvdash \varphi) \to \mathcal{T} \vdash \varphi && \leftrightarrow \mathrm{DNE}(\lambda Xp.\ \mathcal{D}_\mathsf{L}X \wedge \mathcal{E}_\mathsf{L}\,p) \\
&\leftrightarrow \forall \Gamma\varphi. && \neg(\Gamma \nvdash \varphi) \to \Gamma \vdash \varphi && \leftrightarrow \mathrm{DNE}(\lambda Xp.\ p = \lambda \mathsf{s}.\exists \mathsf{t}.\mathsf{s} \triangleright \mathsf{t})
\end{aligned}$$

Theorem

Let $p : X \to \mathbb{P}$ and $q : Y \to \mathbb{P}$. If p is stable and $p \preceq q$ then q is stable, where

$$\exists f : X \to Y.\ \forall x.\ px \leftrightarrow q(fx).$$

Undecidability: The Entscheidungsproblem*

*F., K., and Gert Smolka at CPP'19.

General Idea

Conventional outline following Turing:

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

General Idea

Conventional outline following Turing:

- Encode Turing machine M as formula φ_M over custom signature
- Verify that M halts if and only if φ_M holds in all models
- Verify that M halts if and only if φ_M is provable in intuitionistic natural deduction
- Verify that M halts if and only if φ_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*

$\frac{C2}{LoC2018}$	$\frac{xfor}{}$	$\frac{nf}{d}$	$\frac{FLo}{F}$	$\frac{d}{ord}$	$\frac{018inO}{inOxf}$

*Post (1946); Forster et al. (2018)

The Post Correspondence Problem PCP*

$$\boxed{\frac{C2}{LoC2018}}\ \boxed{\frac{xfor}{}}\ \boxed{\frac{nf}{d}}\ \boxed{\frac{FLo}{F}}\ \boxed{\frac{d}{ord}}\ \boxed{\frac{018inO}{inOxf}}$$

$$\boxed{\frac{FLo}{F}}$$

*Post (1946); Forster et al. (2018)

The Post Correspondence Problem PCP*

$\frac{C2}{LoC2018}$ $\frac{xfor}{}$ $\frac{nf}{d}$ $\frac{FLo}{F}$ $\frac{d}{ord}$ $\frac{018inO}{inOxf}$

$\frac{FLo}{F}$ $\frac{C2}{LoC2018}$

*Post (1946); Forster et al. (2018)

The Post Correspondence Problem PCP*

$\boxed{\frac{C2}{LoC2018}}$ $\boxed{\frac{xfor}{}}$ $\boxed{\frac{nf}{d}}$ $\boxed{\frac{FLo}{F}}$ $\boxed{\frac{d}{ord}}$ $\boxed{\frac{018inO}{inOxf}}$

$\boxed{\frac{FLo}{F}}$ $\boxed{\frac{C2}{LoC2018}}$ $\boxed{\frac{018inO}{inOxf}}$

*Post (1946); Forster et al. (2018)

The Post Correspondence Problem PCP*

$\boxed{\dfrac{C2}{LoC2018}}\ \boxed{\dfrac{xfor}{}}\ \boxed{\dfrac{nf}{d}}\ \boxed{\dfrac{FLo}{F}}\ \boxed{\dfrac{d}{ord}}\ \boxed{\dfrac{018inO}{inOxf}}$

$\boxed{\dfrac{FLo}{F}}\ \boxed{\dfrac{C2}{LoC2018}}\ \boxed{\dfrac{018inO}{inOxf}}\ \boxed{\dfrac{xfor}{}}$

*Post (1946); Forster et al. (2018)

The Post Correspondence Problem PCP*

$$\boxed{\frac{C2}{LoC2018}}\ \boxed{\frac{xfor}{}}\ \boxed{\frac{nf}{d}}\ \boxed{\frac{FLo}{F}}\ \boxed{\frac{d}{ord}}\ \boxed{\frac{018inO}{inOxf}}$$

$$\boxed{\frac{FLo}{F}}\ \boxed{\frac{C2}{LoC2018}}\ \boxed{\frac{018inO}{inOxf}}\ \boxed{\frac{xfor}{}}\ \boxed{\frac{d}{ord}}$$

*Post (1946); Forster et al. (2018)

The Post Correspondence Problem PCP*

$$\boxed{\frac{C2}{LoC2018}}\ \boxed{\frac{xfor}{}}\ \boxed{\frac{nf}{d}}\ \boxed{\frac{FLo}{F}}\ \boxed{\frac{d}{ord}}\ \boxed{\frac{018inO}{inOxf}}$$

$$\boxed{\frac{FLo}{F}}\ \boxed{\frac{C2}{LoC2018}}\ \boxed{\frac{018inO}{inOxf}}\ \boxed{\frac{xfor}{}}\ \boxed{\frac{d}{ord}}$$

$$\frac{FLoC2018inOxford}{FLoC2018inOxford}$$

*Post (1946); Forster et al. (2018)

The Post Correspondence Problem PCP*

$$\boxed{\frac{C2}{LoC2018}}\ \boxed{\frac{xfor}{}}\ \boxed{\frac{nf}{d}}\ \boxed{\frac{FLo}{F}}\ \boxed{\frac{d}{ord}}\ \boxed{\frac{018inO}{inOxf}}$$

$$\boxed{\frac{FLo}{F}}\ \boxed{\frac{C2}{LoC2018}}\ \boxed{\frac{018inO}{inOxf}}\ \boxed{\frac{xfor}{}}\ \boxed{\frac{d}{ord}}$$

$$\frac{FLoC2018inOxford}{FLoC2018inOxford}$$

Base type: $\mathbb{L}(\mathbb{LB} \times \mathbb{LB})$
Definition: $\mathsf{PCP}(L) := \exists x : \mathbb{LB}.\ L \triangleright (x, x)$

$$\frac{(u, v) \in L}{L \triangleright (u, v)} \qquad \frac{L \triangleright (x, y) \quad (u, v) \in L}{L \triangleright (x \mathbin{+\!\!+} u, y \mathbin{+\!\!+} v)}$$

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. $\overline{\mathsf{tt}\,\mathsf{ff}\,\mathsf{ff}\,\mathsf{tt}} = f_{\mathsf{tt}}\,(f_{\mathsf{ff}}\,(f_{\mathsf{ff}}\,(f_{\mathsf{tt}}\,(e))))$.

A Standard Model

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

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:

$$e^{\mathcal{B}} := [] \qquad\qquad Q^{\mathcal{B}} := \mathsf{PCP}\,S$$
$$f_b^{\mathcal{B}}\,s := b :: s \qquad\qquad P^{\mathcal{B}}\,s\,t := S \triangleright s/t.$$

A Standard Model

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

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:

$$e^{\mathcal{B}} := [] \qquad\qquad Q^{\mathcal{B}} := \mathsf{PCP}\, S$$

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

Lemma

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

The reduction

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

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

$$\varphi_S := \varphi_1 \dot\to \varphi_2 \dot\to \varphi_3 \dot\to Q$$

The reduction

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

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

$$\varphi_S := \varphi_1 \dot\to \varphi_2 \dot\to \varphi_3 \dot\to Q$$

1. PCP $S \to\; \vdash \varphi_S$
2. $\vdash$ is sound for Tarski semantics w.r.t. all models
3. $\mathcal{B} \vDash \varphi_S \to$ PCP S

Theorem

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

Theorem

$\overline{\text{PCP}}$ *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 φ^Q of formulas φ such that $A \vdash_c \varphi$ implies $A^Q \vdash \varphi^Q$.

Undecidability of Classical Provability

Soundness is not usable! As a remedy, we define a Gödel-Gentzen-Friedman translation φ^Q of formulas φ 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. $\mathcal{B} \vDash \varphi_S^Q \to \mathcal{B} \vDash \varphi_S$

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 $x, y_1, \ldots, y_m$,*

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

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 $s_1{}^1$.$\boxtimes$

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 $\gamma(r, y)$ such that, for $n \geqq 1$,*

$$[r]^A_{1+n}(y, \mathfrak{x}^{(n)}) = [\gamma(r, y)]_n{}^A(\mathfrak{x}^{(n)}).$$

Intuitively, this result may be interpreted, for $A = \phi$, $n = 1$, as declaring the existence of an algorithm[1] by means of which, given any Turing machine Z and number m, a Turing machine Z_m can be found such that

$$\Psi_Z{}^{(2)}(m, x) = \Psi_{Z_m}(x).$$

Now it is clear that there exist Turing machines Z_m satisfying this last relation since, for each fixed m, $\Psi_Z{}^{(2)}(m, x)$ is certainly a partial recursive function of x. Hence, the content of our theorem (in this special case) is that Z_m can be found effectively in terms of Z and m. However, such a Z_m can readily be described as a Turing machine which, beginning at $\alpha = q_1 1^{x+1}$, proceeds to print $\bar{m} = 1^{m+1}$ to the left, eventually arriving at $\beta = q_N 1^{m+1} B 1^{x+1}$, and then proceeds to act like Z when confronted with

[1] Actually, an algorithm given by a *primitive* recursive function.

$q_1 1^{m+1} B 1^{x+1}$. As the general case does not differ essentially from this special case, all that is required for a formal proof is a detailed construction of Z_m and a careful consideration of the Gödel numbers. The reader who wishes to omit the tedious details, and simply accept the result, may well do so.

PROOF OF THEOREM 1.1. For each value of y, let W_y be the Turing machine consisting of the following quadruples:

$$\begin{array}{l} q_1\ 1\ L\ q_1 \\ q_1\ B\ L\ q_2 \\ \left.\begin{array}{l} q_{i+1}\ B\ 1\ q_{i+1} \\ q_{i+1}\ 1\ L\ q_{i+2} \end{array}\right\} 1 \leqq i \leqq y \\ q_{y+2}\ B\ 1\ q_{y+3}. \end{array}$$

Then, with respect to W_y,

$$\begin{aligned} q_1(\overline{\mathfrak{x}^{(n)}}) &\to q_1 B(\overline{\mathfrak{x}^{(n)}}) \\ &\to q_2 BB(\overline{\mathfrak{x}^{(n)}}) \\ &\to \cdots \\ &\to q_{y+2}(\overline{y, \mathfrak{x}^{(n)}}). \end{aligned}$$

Let r be a Gödel number of a Turing machine Z, and let

$$Z_y = W_y \cup Z^{(y+2)}.\dagger$$

Then, since the quadruples of $Z^{(y+2)}$ have precisely the same effect on $q_{y+2}(\overline{y, \mathfrak{x}^{(n)}})$ that those of Z have on $q_1(\overline{y, \mathfrak{x}^{(n)}})$, we have

$$\Psi^{(n)}_{Z_y;A}(\mathfrak{x}^{(n)}) = \Psi_Z^{(1+n)}(y, \mathfrak{x}^{(n)}) = [r]^A_{1+n}(y, \mathfrak{x}^{(n)}). \tag{1}$$

We now proceed to evaluate one of the Gödel numbers of Z_y as a function of r and y. The Gödel numbers of the quadruples that make up W_y are as follows:[1]

$$\begin{aligned} a &= \mathrm{gn}\ (q_1\ 1\ L\ q_1) = 2^9 \cdot 3^{11} \cdot 5^5 \cdot 7^9, \\ b &= \mathrm{gn}\ (q_1\ B\ L\ q_2) = 2^9 \cdot 3^7 \cdot 5^5 \cdot 7^{13}, \\ c(i) &= \mathrm{gn}\ (q_{i+1}\ B\ 1\ q_{i+1}) = 2^{4i+9} \cdot 3^7 \cdot 5^{11} \cdot 7^{4i+9}, \ 1 \leqq i \leqq y, \\ d(i) &= \mathrm{gn}\ (q_{i+1}\ 1\ L\ q_{i+2}) = 2^{4i+9} \cdot 3^{11} \cdot 5^5 \cdot 7^{4i+13}, \ 1 \leqq i \leqq y, \\ e(y) &= \mathrm{gn}\ (q_{y+2}\ B\ 1\ q_{i+3}) = 2^{4y+13} \cdot 3^7 \cdot 5^{11} \cdot 7^{4y+17}. \end{aligned}$$

Thus, if we let

$$\varphi(y) = 2^a \cdot 3^b \cdot 5^{e(y)} \cdot \prod_{i=1}^{y} [\mathrm{Pr}\ (i + 3)^{c(i)}\ \mathrm{Pr}\ (i + y + 3)^{d(i)}],$$

then $\varphi(y)$ is a primitive recursive function, and, for each y, $\varphi(y)$ is a Gödel number of W_y.

We recall that the predicate IC (x), which is true if and only if x is the number associated with an internal configuration q_i, is primitive recursive, since

$$\mathrm{IC}\ (x) \leftrightarrow \bigvee_{y=0}^{x} (x = 4y + 9).$$

Hence, the function $\iota(x)$, which is 1 when x is the number associated with a q_i and 0 otherwise, is primitive recursive. If h is the Gödel number of a quadruple, then the Gödel number of the quadruple obtained from this one by replacing each q_i by q_{i+y+2} is

$$f(h, y) = 2^{1\,\mathrm{Gl}\,h+4y+8} \cdot 3^{2\,\mathrm{Gl}\,h} \cdot 5^{3\,\mathrm{Gl}\,h+(4y+8)\iota(3\,\mathrm{Gl}\,h)} \cdot 7^{4\,\mathrm{Gl}\,h+4y+8}.$$

Here, $f(h, y)$ is primitive recursive. Hence, if we let

$$\theta(r, y) = \prod_{i=1}^{\mathfrak{L}(r)} \mathrm{Pr}\ (i)^{f(i\,\mathrm{Gl}\,r, y)},$$

then $\theta(r, y)$ is a primitive recursive function and, for each y, $\theta(r, y)$ is a Gödel number of $Z^{(y+2)}$.

Let $\tau(x) = 1$ if x is a Gödel number of a Turing machine; 0, otherwise. Then, by (11) of Chap. 4, Sec. 1, $\tau(x)$ is primitive recursive. Finally, let

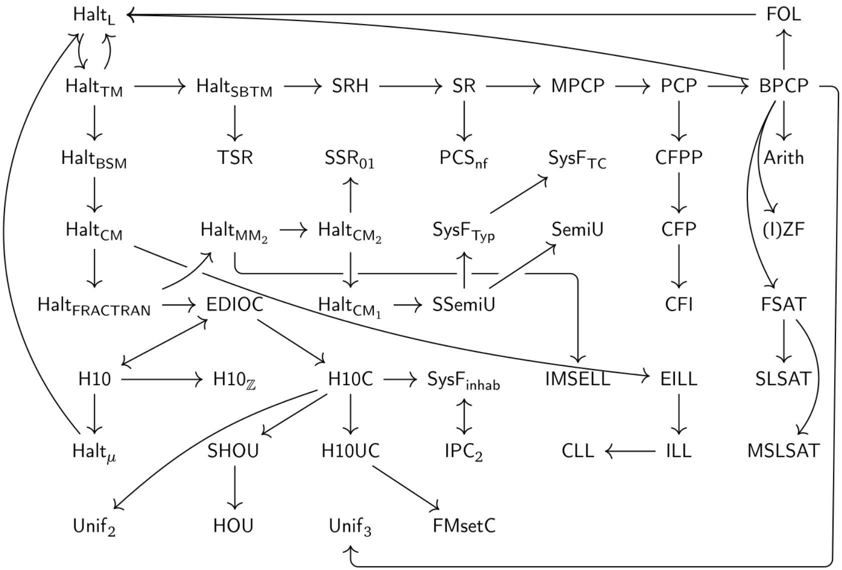
$$\gamma(r, y) = (\varphi(y) * \theta(r, y))\tau(r).$$

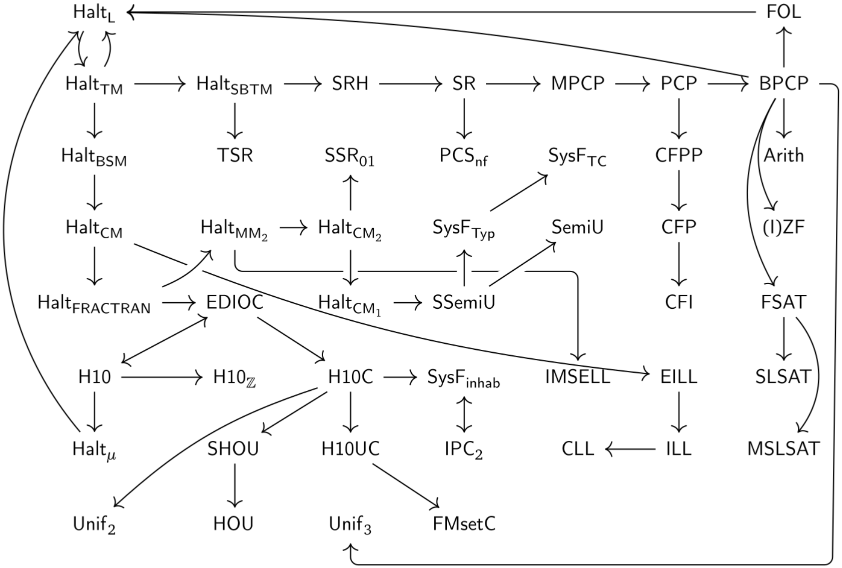
Then $\gamma(r, y)$ is a primitive recursive function and, for each y, $\gamma(r, y)$ is a Gödel number of Z_y. Hence, by (1),

$$[\gamma(r, y)]_n{}^A(\mathfrak{x}^{(n)}) = [r]^A_{1+n}(y, \mathfrak{x}^{(n)}). \tag{2}$$

It remains only to consider the case where r is not a Gödel number of a Turing machine. In that case, $\gamma(r, y)$, as defined above, is 0 and, thus, is itself not the Gödel number of a Turing machine; so (2) remains correct.[1]

```
section ‹The $s$-$m$-$n$ theorem›

text ‹For all $m, n > 0$ there is an $(m + 1)$-ary primitive recursive
function $s^m_n$ with
\[
  \varphi_p^{(m + n)}(c_1, \dots,c_m, x_1, \dots, x_n) =
  \varphi_{s^m_n(p, c_1, \dots,c_m)}^{(n)}(x_1, \dots, x_n)
\]
for all $p, c_1, \ldots, c_m, x_1, \ldots, x_n$. Here, $\varphi^{(n)}$ is a
function universal for $n$-ary partial recursive functions, which we will
represent by @{term "r_universal n"}›

text ‹The $s^m_n$ functions compute codes of functions. We start simple:
computing codes of the unary constant functions.›

fun code_const1 :: "nat ⇒ nat" where
  "code_const1 0 = 0"
| "code_const1 (Suc c) = quad_encode 3 1 1 (singleton_encode (code_const1 c))"

lemma code_const1: "code_const1 c = encode (r_const c)"
  by (induction c) simp_all

definition "r_code_const1_aux ≡
  Cn 3 r_prod_encode
    [r_constn 2 3,
     Cn 3 r_prod_encode
      [r_constn 2 1,
       Cn 3 r_prod_encode
        [r_constn 2 1, Cn 3 r_singleton_encode [Id 3 1]]]]"

lemma r_code_const1_aux_prim: "prim_recfn 3 r_code_const1_aux"
  by (simp_all add: r_code_const1_aux_def)

lemma r_code_const1_aux:
  "eval r_code_const1_aux [i, r, c] ↓= quad_encode 3 1 1 (singleton_encode r)"
  by (simp add: r_code_const1_aux_def)

definition "r_code_const1 ≡ r_shrink (Pr 1 Z r_code_const1_aux)"

lemma r_code_const1_prim: "prim_recfn 1 r_code_const1"
  by (simp_all add: r_code_const1_def r_code_const1_aux_prim)

lemma r_code_const1: "eval r_code_const1 [c] ↓= code_const1 c"
proof -
  let ?h = "Pr 1 Z r_code_const1_aux"
  have "eval ?h [c, x] ↓= code_const1 c" for x
    using r_code_const1_aux r_code_const1_def
    by (induction c) (simp_all add: r_code_const1_aux_prim)
  then show ?thesis by (simp add: r_code_const1_def r_code_const1_aux_prim)
qed

text ‹Functions that compute codes of higher-arity constant functions:›

definition code_constn :: "nat ⇒ nat ⇒ nat" where
  "code_constn n c ≡
    if n = 1 then code_const1 c
    else quad_encode 3 n (code_const1 c) (singleton_encode (triple_encode 2 n 0))"

lemma code_constn: "code_constn (Suc n) c = encode (r_constn n c)"
  unfolding code_constn_def using code_const1 r_constn_def
  by (cases "n = 0") simp_all

definition r_code_constn :: "nat ⇒ recf" where
  "r_code_constn n ≡
     if n = 1 then r_code_const1
     else
       Cn 1 r_prod_encode
        [r_const 3,
         Cn 1 r_prod_encode
          [r_const n,
           Cn 1 r_prod_encode
            [r_code_const1,
             Cn 1 r_singleton_encode
              [Cn 1 r_prod_encode
                [r_const 2, Cn 1 r_prod_encode [r_const n, Z]]]]]]"

lemma r_code_constn_prim: "prim_recfn 1 (r_code_constn n)"
  by (simp_all add: r_code_constn_def r_code_const1_prim)

lemma r_code_constn: "eval (r_code_constn n) [c] ↓= code_constn n c"
  by (auto simp add: r_code_constn_def r_code_const1 code_constn_def r_code_const1_prim)

text ‹Computing codes of $m$-ary projections:›

definition code_id :: "nat ⇒ nat ⇒ nat" where
  "code_id m n ≡ triple_encode 2 m n"

lemma code_id: "encode (Id m n) = code_id m n"
  unfolding code_id_def by simp

text ‹The functions $s^m_n$ are represented by the following function.
The value $m$ corresponds to the length of @{term "cs"}.›

definition smn :: "nat ⇒ nat ⇒ nat list ⇒ nat" where
  "smn n p cs ≡ quad_encode
     3
     n
     (encode (r_universal (n + length cs)))
     (list_encode (code_constn n p # map (code_constn n) cs @ map (code_id n) [0..<n]))"

lemma smn:
  assumes "n > 0"
  shows "smn n p cs = encode
   (Cn n
     (r_universal (n + length cs))
     (r_constn (n - 1) p # map (r_constn (n - 1)) cs @ (map (Id n) [0..<n])))"
proof -
  let ?p = "r_constn (n - 1) p"
  let ?gs1 = "map (r_constn (n - 1)) cs"
  let ?gs2 = "map (Id n) [0..<n]"
  let ?gs = "?p # ?gs1 @ ?gs2"
  have "map encode ?gs1 = map (code_constn n) cs"
    by (intro nth_equalityI; auto; metis code_constn assms Suc_pred)
  moreover have "map encode ?gs2 = map (code_id n) [0..<n]"
    by (rule nth_equalityI) (auto simp add: code_id_def)
  moreover have "encode ?p = code_constn n p"
    using assms code_constn[of "n - 1" p] by simp
  ultimately have "map encode ?gs =
      code_constn n p # map (code_constn n) cs @ map (code_id n) [0..<n]"
    by simp
  then show ?thesis
    unfolding smn_def using assms encode.simps(4) by presburger
qed

text ‹The next function is to help us define @{typ recf}s corresponding

    list_encode (map (code_constn n) (p # cs) @ map (code_id n) [0..<n])"
proof -
  let ?xs = "map (λi. Cn (Suc m) (r_code_constn n) [Id (Suc m) i]) [0..<Suc m]"
  let ?ys = "map (λi. r_constn m (code_id n i)) [0..<n]"
  have len_xs: "length ?xs = Suc m" by simp

  have map_xs: "map (λg. eval g (p # cs)) ?xs = map Some (map (code_constn n) (p # cs))"
  proof (intro nth_equalityI)
    show len: "length (map (λg. eval g (p # cs)) ?xs) =
        length (map Some (map (code_constn n) (p # cs)))"
      by (simp add: assms(2))

    have "map (λg. eval g (p # cs)) ?xs ! i = map Some (map (code_constn n) (p # cs)) ! i"
      if "i < Suc m" for i
    proof -
      have "map (λg. eval g (p # cs)) ?xs ! i = (λg. eval g (p # cs)) (?xs ! i)"
        using len_xs that by (metis nth_map)
      also have "... = eval (Cn (Suc m) (r_code_constn n) [Id (Suc m) i]) (p # cs)"
        using that len_xs
        by (metis (no_types, lifting) add.left_neutral length_map nth_map nth_upt)
      also have "... = eval (r_code_constn n) [the (eval (Id (Suc m) i) (p # cs))]"
        using r_code_constn_prim assms(2) that by simp
      also have "... = eval (r_code_constn n) [(p # cs) ! i]"
        using len that by simp
      finally have "map (λg. eval g (p # cs)) ?xs ! i ↓= code_constn n ((p # cs) ! i)"
        using r_code_constn by simp
      then show ?thesis
        using len_xs len that by (metis length_map nth_map)
    qed
    moreover have "length (map (λg. eval g (p # cs)) ?xs) = Suc m" by simp
    ultimately show "⋀i. i < length (map (λg. eval g (p # cs)) ?xs) ⟹
        map (λg. eval g (p # cs)) ?xs ! i =
        map Some (map (code_constn n) (p # cs)) ! i"
      by simp
  qed
  moreover have "map (λg. eval g (p # cs)) ?ys = map Some (map (code_id n) [0..<n])"
    using assms(2) by (intro nth_equalityI; auto)
  ultimately have "map (λg. eval g (p # cs)) (?xs @ ?ys) =
      map Some (map (code_constn n) (p # cs) @ map (code_id n) [0..<n])"
    by (metis map_append)
  moreover have "map (λx. the (eval x (p # cs))) (?xs @ ?ys) =
      map the (map (λx. eval x (p # cs)) (?xs @ ?ys))"
    by simp
  ultimately have *: "map (λg. the (eval g (p # cs))) (?xs @ ?ys) =
      (map (code_constn n) (p # cs) @ map (code_id n) [0..<n])"
    by simp

  have "∀i<length ?xs. eval (?xs ! i) (p # cs) = map (λg. eval g (p # cs)) ?xs ! i"
    by (metis nth_map)
  then have
    "∀i<length ?xs. eval (?xs ! i) (p # cs) = map Some (map (code_constn n) (p # cs)) ! i"
    using map_xs by simp
  then have "∀i<length ?xs. eval (?xs ! i) (p # cs) ↓"
    using assms map_xs by (metis length_map nth_map option.simps(3))
  then have xs_converg: "∀z∈set ?xs. eval z (p # cs) ↓"
    by (metis in_set_conv_nth)

  have "∀i<length ?ys. eval (?ys ! i) (p # cs) = map (λx. eval x (p # cs)) ?ys ! i"
    by simp
  then have
    "∀i<length ?ys. eval (?ys ! i) (p # cs) = map Some (map (code_id n) [0..<n]) ! i"
    using assms(2) by simp
  then have "∀i<length ?ys. eval (?ys ! i) (p # cs) ↓"
    by simp
  then have "∀z∈set (?xs @ ?ys). eval z (p # cs) ↓"
    using xs_converg by auto
  moreover have "recfn (length (p # cs)) (Cn (Suc m) (r_list_encode (m + n)) (?xs @ ?ys))"
    using assms r_code_constn_prim by auto
  ultimately have "eval (r_smn_aux n m) (p # cs) =
      eval (r_list_encode (m + n)) (map (λg. the (eval g (p # cs))) (?xs @ ?ys))"
    unfolding r_smn_aux_def using assms by simp
  then have "eval (r_smn_aux n m) (p # cs) =
      eval (r_list_encode (m + n)) (map (code_constn n) (p # cs) @ map (code_id n) [0..<n])"
    using * by metis
  moreover have "length (?xs @ ?ys) = Suc (m + n)" by simp
  ultimately show ?thesis
    using r_list_encode * assms(1) by (metis (no_types, lifting) length_map)
qed

text ‹For all $m, n > 0$, the @{typ recf} corresponding to $s^m_n$ is
given by the next function.›

definition r_smn :: "nat ⇒ nat ⇒ recf" where
 "r_smn n m ≡
    Cn (Suc m) r_prod_encode
     [r_constn m 3,
      Cn (Suc m) r_prod_encode
       [r_constn m n,
        Cn (Suc m) r_prod_encode
         [r_constn m (encode (r_universal (n + m))), r_smn_aux n m]]]"

lemma r_smn_prim [simp]: "n > 0 ⟹ prim_recfn (Suc m) (r_smn n m)"
  by (simp_all add: r_smn_def r_smn_aux_prim)

lemma r_smn:
  assumes "n > 0" and "length cs = m"
  shows "eval (r_smn n m) (p # cs) ↓= smn n p cs"
  using assms r_smn_def r_smn_aux smn_def r_smn_aux_prim by simp

lemma map_eval_Some_the:
  assumes "map (λg. eval g xs) gs = map Some ys"
  shows "map (λg. the (eval g xs)) gs = ys"
  using assms
  by (metis (no_types, lifting) length_map nth_equalityI nth_map option.sel)

text ‹The essential part of the $s$-$m$-$n$ theorem: For all $m, n > 0$
the function $s^m_n$ satisfies
\[
  \varphi_p^{(m + n)}(c_1, \dots,c_m, x_1, \dots, x_n) =
  \varphi_{s^m_n(p, c_1, \dots,c_m)}^{(n)}(x_1, \dots, x_n)
\] for all $p, c_i, x_j$.›

lemma smn_lemma:
  assumes "n > 0" and len_cs: "length cs = m" and len_xs: "length xs = n"
  shows "eval (r_universal (m + n)) (p # cs @ xs) =
    eval (r_universal n) ((the (eval (r_smn n m) (p # cs))) # xs)"
proof -
  let ?s = "r_smn n m"
  let ?f = "Cn n
    (r_universal (n + length cs))
    (r_constn (n - 1) p # map (r_constn (n - 1)) cs @ (map (Id n) [0..<n]))"
  have "eval ?s (p # cs) ↓= smn n p cs"
    using assms r_smn by simp
  then have eval_s: "eval ?s (p # cs) ↓= encode ?f"
    by (simp add: assms(1) smn)

      show "length (map (λg. the (eval g xs)) ?gs) = length (p # cs @ xs)"
        by (simp add: len_xs)
      have len: "length (map (λg. the (eval g xs)) ?gs) = Suc (m + n)"
        by (simp add: len_cs)
      moreover have "map (λg. the (eval g xs)) ?gs ! i = (p # cs @ xs) ! i"
        if "i < Suc (m + n)" for i
      proof -
        from that consider "i = 0" | "i > 0 ∧ i < Suc m" | "Suc m ≤ i ∧ i < Suc (m + n)"
          using not_le_imp_less by auto
        then show ?thesis
        proof (cases)
          case 1
          then show ?thesis using assms(1) len_xs by simp
        next
          case 2
          then have "?gs ! i = (map (r_constn (n - 1)) cs) ! (i - 1)"
            using len_cs
            by (metis One_nat_def Suc_less_eq Suc_pred length_map
              less_numeral_extra(3) nth_Cons' nth_append)
          then have "map (λg. the (eval g xs)) ?gs ! i =
              (λg. the (eval g xs)) ((map (r_constn (n - 1)) cs) ! (i - 1))"
            using len by (metis length_map nth_map that)
          also have "... = the (eval ((r_constn (n - 1) (cs ! (i - 1)))) xs)"
            using 2 len_cs by auto
          also have "... = cs ! (i - 1)"
            using r_constn len_xs assms(1) by simp
          also have "... = (p # cs @ xs) ! i"
            using 2 len_cs
            by (metis diff_Suc_1 less_Suc_eq_0_disj less_numeral_extra(3) nth_Cons' nth_append)
          finally show ?thesis .
        next
          case 3
          then have "?gs ! i = (map (Id n) [0..<n]) ! (i - Suc m)"
            using len_cs
            by (simp; metis (no_types, lifting) One_nat_def Suc_less_eq add_leE
              plus_1_eq_Suc diff_diff_left length_map not_le nth_append
              ordered_cancel_comm_monoid_diff_class.add_diff_inverse)
          then have "map (λg. the (eval g xs)) ?gs ! i =
              (λg. the (eval g xs)) ((map (Id n) [0..<n]) ! (i - Suc m))"
            using len by (metis length_map nth_map that)
          also have "... = the (eval ((Id n (i - Suc m))) xs)"
            using 3 len_cs by auto
          also have "... = xs ! (i - Suc m)"
            using len_xs 3 by auto
          also have "... = (p # cs @ xs) ! i"
            using len_cs len_xs 3
            by (metis diff_Suc_1 diff_diff_left less_Suc_eq_0_disj not_le nth_Cons'
              nth_append plus_1_eq_Suc)
          finally show ?thesis .
        qed
      qed
      ultimately show "map (λg. the (eval g xs)) ?gs ! i = (p # cs @ xs) ! i"
        if "i < length (map (λg. the (eval g xs)) ?gs)" for i
        using that by simp
    qed
    ultimately show ?thesis by simp
qed

theorem smn_theorem:
  assumes "n > 0"
  shows "∃s. prim_recfn (Suc m) s ∧
    (∀p cs xs. length cs = m ∧ length xs = n ⟶
        eval (r_universal (m + n)) (p # cs @ xs) =
        eval (r_universal n) ((the (eval s (p # cs))) # xs))"
  using smn_lemma exI[of _ "r_smn n m"] assms by simp
```


$\mathsf{Halt_L}$
FOL
$\mathsf{Halt_{TM}}$
$\mathsf{Halt_{SBTM}}$
SRH
SR
MPCP
PCP
BPCP
$\mathsf{Halt_{BSM}}$
TSR
$\mathsf{SSR_{01}}$
$\mathsf{PCS_{nf}}$
$\mathsf{SysF_{TC}}$
CFPP
Arith
$\mathsf{Halt_{CM}}$
$\mathsf{Halt_{MM_2}}$
$\mathsf{Halt_{CM_2}}$
$\mathsf{SysF_{Typ}}$
SemiU
CFP
(I)ZF
$\mathsf{Halt_{FRACTRAN}}$
EDIOC
$\mathsf{Halt_{CM_1}}$
SSemiU
CFI
FSAT
H10
$\mathsf{H10_{\mathbb{Z}}}$
H10C
$\mathsf{SysF_{inhab}}$
IMSELL
EILL
SLSAT
$\mathsf{Halt_{\mu}}$
SHOU
H10UC
$\mathsf{IPC_2}$
CLL
ILL
MSLSAT
$\mathsf{Unif_2}$
HOU
$\mathsf{Unif_3}$
FMsetC



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
$$\mathcal{U}(p) := \mathcal{D}p \to \mathcal{E}(\overline{\mathsf{Halt}})$$
- We can prove usual undecidability by extracting reduction functions to L
- We can develop synthetic computability theory based on axiom CT_L stating that $\forall f : \mathbb{N} \to \mathbb{N}. \exists t.$ *the* L*-term t computes f* *

*Kreisel (1965); Forster (2021, 2022)

More Constructive Reverse Mathematics *

*F., K., and W. in Journal of Logic and Computation.

A very confusing literature review

1. Diener (2020) (in Bishop style constructive math):
   Compactness is equivalent to Weak König's Lemma for decidable trees $\mathsf{WKL}_{\mathcal{D}}$
   $\mathsf{WKL}_{\mathcal{D}}$ is equivalent to the fan theorem $\mathsf{FAN}_{\mathcal{D}}$
2. Simpson (1985) (in classical reverse mathematics, RCA_0):
   The model existence theorem is equivalent to WKL
3. Krivtsov (2015) (in Bishop style constructive math):
   Completeness of intuitionistic FOL w.r.t. Kripke semantics is equivalent to $\mathsf{FAN}_{\mathcal{D}}$

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))

$$\mathsf{WKL} \leftrightarrow \mathsf{LLPO} \wedge \Pi^0_1\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$$

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 $\mathcal{D}(\lambda P v.\ \hat{P} v)$ (predicate interpretations are boolean functions).

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

Models matter

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

Theorem

The following are equivalent:

1. *Completeness of $\mathcal{T} \vdash_c \varphi$ 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.*

Models matter

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

Theorem

The following are equivalent:

1. *Completeness of $\mathcal{T} \vdash_c \varphi$ 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.*

$\mathsf{WKL}_{\mathcal{D}}$:= Every decidable and infinite binary tree has an infinite path

Corollary

Compactness for decidable models implies $\mathsf{WKL}_{\mathcal{D}}$.

Open questions

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

$$\mathsf{WKL}_{\mathcal{D}} \leftrightarrow \forall R : \mathbb{N} \to \mathbb{B} \to \mathbb{P}.\ \Pi^0_1 R \to (\forall n. \neg\neg\exists b. Rnb) \to \exists f : \mathbb{N} \to \mathbb{B}. \forall n.\ R\ n\ (fn)$$

Undecidability: Trakhtenbrot's Theorem*

*K. and Dominique Larchey-Wendling at IJCAR'20.

General idea

Given a FOL formula φ, is φ finitely satisfiable?

Textbook proofs by dual reduction from the halting problem:*

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

*e.g. Libkin (2010); Börger et al. (1997)

General idea

Given a FOL formula φ, is φ finitely satisfiable?

Textbook proofs by dual reduction from the halting problem:*

- Encode Turing machine M as formula φ_M over custom signature
- Verify that the models of φ_M correspond to the runs of M
- Conclude that M halts if and only if φ_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 $\mathbb{F}_n$

$\mathsf{FSAT}(\Sigma)\,\varphi$ if additionally D is finite and all $P^{\mathcal{M}}$ are decidable

$\mathsf{FSATEQ}(\Sigma;\equiv)\,\varphi$ if $x \equiv^{\mathcal{M}} y \leftrightarrow x = y$ for all $x, y : D$ (hence discrete)

Encoding the Post Correspondence Problem

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

Encoding the Post Correspondence Problem

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

- Chains like $f_{\mathsf{ff}}(f_{\mathsf{tt}}(e))$ represent strings while $\star$ signals overflow

Encoding the Post Correspondence Problem

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

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

$$\begin{aligned}
\varphi_P &:= \dot\forall xy.\, P\,x\,y \mathrel{\dot\to} x \not\equiv \star \mathrel{\dot\wedge} y \not\equiv \star \\
\varphi_\prec &:= (\dot\forall x.\, x \not\prec x) \mathrel{\dot\wedge} (\dot\forall xyz.\, x \prec y \mathrel{\dot\to} y \prec z \mathrel{\dot\to} x \prec z)
\end{aligned}$$

Encoding the Post Correspondence Problem

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

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

$$
\begin{aligned}
\varphi_P &:= \dot\forall xy.\, P\,x\,y \mathrel{\dot\to} x \not\equiv \star \mathrel{\dot\wedge} y \not\equiv \star \\
\varphi_\prec &:= (\dot\forall x.\, x \not\prec x) \mathrel{\dot\wedge} (\dot\forall xyz.\, x \prec y \mathrel{\dot\to} y \prec z \mathrel{\dot\to} x \prec z)
\end{aligned}
$$

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

$$
\varphi_f := \begin{pmatrix} f_{\mathsf{tt}}\,\star \equiv \star \mathrel{\dot\wedge} f_{\mathsf{ff}}\,\star \equiv \star \\ \dot\forall x.\, f_{\mathsf{tt}}\,x \not\equiv e \\ \dot\forall x.\, f_{\mathsf{ff}}\,x \not\equiv e \end{pmatrix} \mathrel{\dot\wedge} \begin{pmatrix} \dot\forall xy.\, f_{\mathsf{tt}}\,x \not\equiv \star \mathrel{\dot\to} f_{\mathsf{tt}}\,x \equiv f_{\mathsf{tt}}\,y \mathrel{\dot\to} x \equiv y \\ \dot\forall xy.\, f_{\mathsf{ff}}\,x \not\equiv \star \mathrel{\dot\to} f_{\mathsf{ff}}\,x \equiv f_{\mathsf{ff}}\,y \mathrel{\dot\to} x \equiv y \\ \dot\forall xy.\, f_{\mathsf{tt}}\,x \equiv f_{\mathsf{ff}}\,y \mathrel{\dot\to} f_{\mathsf{tt}}\,x \equiv \star \mathrel{\dot\wedge} f_{\mathsf{ff}}\,y \equiv \star \end{pmatrix}
$$

Trakhtenbrot's Theorem

Given an instance R of PCP, we construct a formula φ_R by:

$$\varphi_R := \varphi_P \mathbin{\dot\wedge} \varphi_\prec \mathbin{\dot\wedge} \varphi_f \mathbin{\dot\wedge} \varphi_\triangleright \mathbin{\dot\wedge} \dot\exists x.\, P\,x\,x$$

Trakhtenbrot's Theorem

Given an instance R of PCP, we construct a formula φ_R by:

$$\varphi_R := \varphi_P \mathrel{\dot\wedge} \varphi_\prec \mathrel{\dot\wedge} \varphi_f \mathrel{\dot\wedge} \varphi_\triangleright \mathrel{\dot\wedge} \dot\exists x.\, P\,x\,x$$

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

$$\varphi_\triangleright := \dot\forall xy.\, P\,x\,y \mathrel{\dot\to} \dot{\bigvee_{(s,t)\in R}} \mathrel{\dot\vee} \begin{cases} x \equiv \overline{s} \mathrel{\dot\wedge} y \equiv \overline{t} \\ \dot\exists uv.\, P\,u\,v \mathrel{\dot\wedge} x \equiv \overline{s}u \mathrel{\dot\wedge} y \equiv \overline{t}v \mathrel{\dot\wedge} u/v \prec x/y \end{cases}$$

Trakhtenbrot's Theorem

Given an instance R of PCP, we construct a formula φ_R by:

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

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

$$\varphi_\triangleright := \dot\forall xy.\, P\,x\,y \mathbin{\dot\to} \dot{\bigvee_{(s,t)\in R}} \mathbin{\dot\lor} \begin{cases} x \equiv \overline{s} \mathbin{\dot\land} y \equiv \overline{t} \\ \dot\exists uv.\, P\,u\,v \mathbin{\dot\land} x \equiv \overline{s}u \mathbin{\dot\land} y \equiv \overline{t}v \mathbin{\dot\land} u/v \prec x/y \end{cases}$$

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 φ_R by:

$$\varphi_R := \varphi_P \mathbin{\dot\wedge} \varphi_\prec \mathbin{\dot\wedge} \varphi_f \mathbin{\dot\wedge} \varphi_\triangleright \mathbin{\dot\wedge} \dot\exists x.\, P\,x\,x$$

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

$$\varphi_\triangleright := \dot\forall xy.\, P\,x\,y \mathbin{\dot\rightarrow} \dot{\bigvee_{(s,t)\in R}} \mathbin{\dot\vee} \begin{cases} x \equiv \overline{s} \mathbin{\dot\wedge} y \equiv \overline{t} \\ \dot\exists uv.\, P\,u\,v \mathbin{\dot\wedge} x \equiv \overline{s}u \mathbin{\dot\wedge} y \equiv \overline{t}v \mathbin{\dot\wedge} u/v \prec x/y \end{cases}$$

Theorem

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

Proof.

Trakhtenbrot's Theorem

Given an instance R of PCP, we construct a formula φ_R by:

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

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

$$\varphi_\triangleright := \dot\forall xy.\, P\,x\,y \mathrel{\dot\to} \dot{\bigvee_{(s,t)\in R}} \mathrel{\dot\lor} \begin{cases} x \equiv \overline{s} \mathrel{\dot\land} y \equiv \overline{t} \\ \dot\exists uv.\, P\,u\,v \mathrel{\dot\land} x \equiv \overline{s}u \mathrel{\dot\land} y \equiv \overline{t}v \mathrel{\dot\land} u/v \prec x/y \end{cases}$$

Theorem

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

Proof.

If R has a solution of length n, then φ_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 φ_R by:

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

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

$$\varphi_\triangleright := \dot\forall xy.\, P\,x\,y \mathbin{\dot\to} \dot{\bigvee_{(s,t)\in R}} \mathbin{\dot\lor} \begin{cases} x \equiv \overline{s} \mathbin{\dot\land} y \equiv \overline{t} \\ \dot\exists uv.\, P\,u\,v \mathbin{\dot\land} x \equiv \overline{s}u \mathbin{\dot\land} y \equiv \overline{t}v \mathbin{\dot\land} u/v \prec x/y \end{cases}$$

Theorem

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

Proof.

If R has a solution of length n, then φ_R is satisfied by the model of strings of length bounded by n.
Conversely, if $\mathcal{M} \vDash_\rho \varphi_R$ 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 φ_R by:

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

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

$$\varphi_\triangleright := \dot\forall xy.\, P\,x\,y \mathrel{\dot\rightarrow} \dot\bigvee_{(s,t)\in R} \dot\lor \begin{cases} x \equiv \overline{s} \mathrel{\dot\land} y \equiv \overline{t} \\ \dot\exists uv.\, P\,u\,v \mathrel{\dot\land} x \equiv \overline{s}u \mathrel{\dot\land} y \equiv \overline{t}v \mathrel{\dot\land} u/v \prec x/y \end{cases}$$

Theorem

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

Proof.

If R has a solution of length n, then φ_R is satisfied by the model of strings of length bounded by n. Conversely, if $\mathcal{M} \vDash_\rho \varphi_R$ we can extract a solution of R from $\varphi_\triangleright$ by well-founded induction on $\prec^{\mathcal{M}}$ (which is applicable since $\mathcal{M}$ is finite). □

Signature Transformations

Given a finite and discrete signature Σ with arities bounded by n, we have:

$$\mathsf{FSATEQ}(\Sigma; \equiv) \preceq \mathsf{FSAT}(\Sigma) \preceq \mathsf{FSAT}(\mathbb{0}; P^{n+2}) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$$

Signature Transformations

Given a finite and discrete signature Σ with arities bounded by n, we have:

$$\mathsf{FSATEQ}(\Sigma; \equiv) \preceq \mathsf{FSAT}(\Sigma) \preceq \mathsf{FSAT}(\mathbb{0}; P^{n+2}) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$$

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

Signature Transformations

Given a finite and discrete signature Σ with arities bounded by n, we have:

$$\mathsf{FSATEQ}(\Sigma; \equiv) \preceq \mathsf{FSAT}(\Sigma) \preceq \mathsf{FSAT}(\mathbb{0}; P^{n+2}) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$$

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

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 Σ with arities bounded by n, we have:

$$\mathsf{FSATEQ}(\Sigma;\equiv) \preceq \mathsf{FSAT}(\Sigma) \preceq \mathsf{FSAT}(\mathbb{0}; P^{n+2}) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$$

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

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

$\mathsf{FSAT}'(\Sigma)\,\varphi$ if $\mathsf{FSAT}(\Sigma)\,\varphi$ on a discrete model

Discrete Models

$\mathsf{FSAT}'(\Sigma)\,\varphi$ if $\mathsf{FSAT}(\Sigma)\,\varphi$ on a discrete model

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

Discrete Models

$\mathsf{FSAT}'(\Sigma)\,\varphi$ if $\mathsf{FSAT}(\Sigma)\,\varphi$ on a discrete model

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

Idea: first-order indistinguishability $x \dot{=} y \;:=\; \forall \varphi \rho.\, \mathcal{M} \vDash_{x \cdot \rho} \varphi \leftrightarrow \mathcal{M} \vDash_{y \cdot \rho} \varphi$

Discrete Models

$\mathsf{FSAT}'(\Sigma)\,\varphi$ if $\mathsf{FSAT}(\Sigma)\,\varphi$ on a discrete model

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

Idea: first-order indistinguishability $x \doteq y \;:=\; \forall \varphi \rho.\, \mathcal{M} \vDash_{x \cdot \rho} \varphi \leftrightarrow \mathcal{M} \vDash_{y \cdot \rho} \varphi$

Lemma

The relation $x \doteq y$ is a decidable congruence for the symbols in Σ.

Discrete Models

$\mathsf{FSAT}'(\Sigma)\,\varphi$ if $\mathsf{FSAT}(\Sigma)\,\varphi$ on a discrete model

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

Idea: first-order indistinguishability $x \doteq y \;:=\; \forall \varphi\rho.\, \mathcal{M} \vDash_{x \cdot \rho} \varphi \leftrightarrow \mathcal{M} \vDash_{y \cdot \rho} \varphi$

Lemma

The relation $x \doteq y$ is a decidable congruence for the symbols in Σ.

Fact

$\mathsf{FSAT}'(\Sigma)\,\varphi$ *iff* $\mathsf{FSAT}(\Sigma)\,\varphi$, *hence in particular* $\mathsf{FSAT}'(\Sigma)\,\varphi \preceq \mathsf{FSAT}(\Sigma)\,\varphi$.

Discrete Models

$\mathsf{FSAT}'(\Sigma)\,\varphi$ if $\mathsf{FSAT}(\Sigma)\,\varphi$ on a discrete model

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

Idea: first-order indistinguishability $x \doteq y \ := \ \forall \varphi\rho.\, \mathcal{M} \vDash_{x\cdot\rho} \varphi \leftrightarrow \mathcal{M} \vDash_{y\cdot\rho} \varphi$

Lemma

The relation $x \doteq y$ is a decidable congruence for the symbols in Σ.

Fact

$\mathsf{FSAT}'(\Sigma)\,\varphi$ *iff* $\mathsf{FSAT}(\Sigma)\,\varphi$, *hence in particular* $\mathsf{FSAT}'(\Sigma)\,\varphi \preceq \mathsf{FSAT}(\Sigma)\,\varphi$.

Proof.

If $\mathcal{M} \vDash_\rho \varphi$ pick $\mathcal{M}'$ to be the quotient of $\mathcal{M}$ under $x \doteq y$. $\square$

Compressing Relations: $\mathsf{FSAT}(\mathbb{0}; P^n) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$

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

Compressing Relations: $\mathsf{FSAT}(\mathbb{0}; P^n) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$

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

So let's play set theory!

Compressing Relations: $\mathsf{FSAT}(\mathbb{O}; P^n) \preceq \mathsf{FSAT}(\mathbb{O}; \in^2)$

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

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

$$(P\,x_1 \dots x_n)'_\in := \text{“}(x_1, \dots, x_n) \in p\text{”} \qquad (\dot\forall z.\,\varphi)'_\in := \dot\forall z.\, z \in d \mathrel{\dot\to} (\varphi)'_\in$$
$$(\varphi \mathbin{\dot\Box} \psi)'_\in := (\varphi)'_\in \mathbin{\dot\Box} (\psi)'_\in \qquad (\dot\exists z.\,\varphi)'_\in := \dot\exists z.\, z \in d \mathbin{\dot\wedge} (\varphi)'_\in$$

Compressing Relations: $\mathsf{FSAT}(\mathbb{0}; P^n) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$

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

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

$$(P\,x_1 \dots x_n)'_\in := \text{“}(x_1, \dots, x_n) \in p\text{”} \qquad (\dot\forall z.\, \varphi)'_\in := \dot\forall z.\, z \in d \mathrel{\dot\rightarrow} (\varphi)'_\in$$

$$(\varphi \mathbin{\dot\Box} \psi)'_\in := (\varphi)'_\in \mathbin{\dot\Box} (\psi)'_\in \qquad (\dot\exists z.\, \varphi)'_\in := \dot\exists z.\, z \in d \mathrel{\dot\wedge} (\varphi)'_\in$$

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

Compressing Relations: $\mathsf{FSAT}(\mathbb{0}; P^n) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$

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

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

$$\begin{aligned} (P\,x_1 \dots x_n)'_\in &:= \text{“}(x_1, \dots, x_n) \in p\text{”} & (\dot\forall z.\,\varphi)'_\in &:= \dot\forall z.\, z \in d \mathrel{\dot\rightarrow} (\varphi)'_\in \\ (\varphi \mathbin{\dot\Box} \psi)'_\in &:= (\varphi)'_\in \mathbin{\dot\Box} (\psi)'_\in & (\dot\exists z.\,\varphi)'_\in &:= \dot\exists z.\, z \in d \mathrel{\dot\wedge} (\varphi)'_\in \end{aligned}$$

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

Fact

$\mathsf{FSAT}(\mathbb{0}; P^n)\,\varphi$ *iff* $\mathsf{FSAT}(\mathbb{0}; \in^2)\,\varphi_\in$, *hence* $\mathsf{FSAT}(\mathbb{0}; P^n) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$.

Compressing Relations: $\mathsf{FSAT}(\mathbb{0}; P^n) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$

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

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

$$\begin{aligned} (P\,x_1 \dots x_n)'_\in &:= \text{“}(x_1, \dots, x_n) \in p\text{”} \qquad & (\dot\forall z.\,\varphi)'_\in &:= \dot\forall z.\, z \in d \mathrel{\dot\rightarrow} (\varphi)'_\in \\ (\varphi \mathbin{\dot\Box} \psi)'_\in &:= (\varphi)'_\in \mathbin{\dot\Box} (\psi)'_\in & (\dot\exists z.\,\varphi)'_\in &:= \dot\exists z.\, z \in d \mathbin{\dot\wedge} (\varphi)'_\in \end{aligned}$$

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

Fact

$\mathsf{FSAT}(\mathbb{0}; P^n)\,\varphi$ *iff* $\mathsf{FSAT}(\mathbb{0}; \in^2)\,\varphi_\in$, *hence* $\mathsf{FSAT}(\mathbb{0}; P^n) \preceq \mathsf{FSAT}(\mathbb{0}; \in^2)$.

Proof.

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

Full Signature Classification

Composing all signature transformations verified we obtain:

Theorem

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

Full Signature Classification

Composing all signature transformations verified we obtain:

Theorem

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

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

Theorem

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

Full Signature Classification

Composing all signature transformations verified we obtain:

Theorem

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

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

Theorem

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

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

Fact

If Σ is discrete and enumerable, then $\mathsf{FSAT}(\Sigma)$ *is enumerable.*

Relativised Entscheidungsproblem and Incompleteness*

*K. and Marc Hermes at ITP'21.

General Idea

Relativised Entscheidungsproblem: is a formula φ entailed by an axiomatisation A?

Strategy if A is strong enough to capture computation:

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

General Idea

Relativised Entscheidungsproblem: is a formula φ entailed by an axiomatisation A?

Strategy if A is strong enough to capture computation:

- Encode Turing machine M as formula φ_M
- Verify that M halts iff $A \vDash \varphi_M$
- Verify that M halts iff $A \vdash \varphi_M$ ($\rightarrow$ 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 $p \preceq A^{\vdash}$ *and there is* x *with* $\neg p\,x$ *then* $A \nvdash \perp$.

Connection of Undecidability to Consistency and Incompleteness

Fact (Consistency)

If $p \preceq A^{\vdash}$ *and there is* x *with* $\neg p\, x$ *then* $A \nvdash \bot$.

Proof.

Let f witness $p \preceq A^{\vdash}$. Then $A \nvdash f\, x$ by $\neg p\, x$ and thus $A \nvdash \bot$. $\square$

Connection of Undecidability to Consistency and Incompleteness

Fact (Consistency)

If $p \preceq A^{\vdash}$ and there is x with $\neg p\,x$ then $A \nvdash \bot$.

Proof.

Let f witness $p \preceq A^{\vdash}$. Then $A \nvdash f\,x$ by $\neg p\,x$ and thus $A \nvdash \bot$. □

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 $p \preceq A^{\vdash}$ and there is x with $\neg p\,x$ then $A \nvdash \bot$.

Proof.

Let f witness $p \preceq A^{\vdash}$. Then $A \nvdash f\,x$ by $\neg p\,x$ and thus $A \nvdash \bot$. □

Fact (Synthetic Incompleteness)

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

Proof.

$A^{\vdash}$ is enumerable and, given completeness and consistency, also co-enumerable as then $A \nvdash \varphi$ iff $A \vdash \neg\varphi$.

Connection of Undecidability to Consistency and Incompleteness

Fact (Consistency)

If $p \preceq A^{\vdash}$ and there is x with $\neg p\,x$ then $A \nvdash \bot$.

Proof.

Let f witness $p \preceq A^{\vdash}$. Then $A \nvdash f\,x$ by $\neg p\,x$ and thus $A \nvdash \bot$. □

Fact (Synthetic Incompleteness)

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

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 $A \vdash \varphi \lor A \nvdash \varphi$. □

Sketch for Peano Arithmetic

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

Sketch for Peano Arithmetic

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

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 $(0, \mathsf{S}, +, \cdot\,; \equiv)$.

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

$L = [c_1, \dots, c_k]$ with maximal index x_n is solvable iff $\mathsf{PA} \vDash \exists^n c_1 \land \dots \land c_k$.

Sketch for Peano Arithmetic

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

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

$L = [c_1, \dots, c_k]$ *with maximal index* x_n *is solvable iff* $\mathsf{PA} \vDash \exists^n c_1 \wedge \dots \wedge c_k$.

Proof.

If L has solution η instantiate the existential quantifiers with numerals $\overline{\eta_1}, \dots, \overline{\eta_n}$. Then the axioms of PA entail the constraints.

Sketch for Peano Arithmetic

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

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

$L = [c_1, \dots, c_k]$ with maximal index x_n is solvable iff $\mathsf{PA} \vDash \exists^n c_1 \wedge \dots \wedge c_k$.

Proof.

If L has solution η instantiate the existential quantifiers with numerals $\overline{\eta_1}, \dots, \overline{\eta_n}$. Then the axioms of PA entail the constraints.

If $\mathsf{PA} \vDash \exists^n c_1 \wedge \dots \wedge c_k$ use the standard model $\mathbb{N}$ to extract solution η. □

Sketch for Peano Arithmetic

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

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

$L = [c_1, \dots, c_k]$ with maximal index x_n is solvable iff $\mathsf{PA} \vDash \exists^n c_1 \wedge \dots \wedge c_k$.

Proof.

If L has solution η instantiate the existential quantifiers with numerals $\overline{\eta_1}, \dots, \overline{\eta_n}$. Then the axioms of PA entail the constraints.

If $\mathsf{PA} \vDash \exists^n c_1 \wedge \dots \wedge c_k$ use the standard model $\mathbb{N}$ to extract solution η. □

Fact

$L = [c_1, \dots, c_k]$ with maximal index x_n is solvable iff $\mathsf{PA} \vdash \exists^n c_1 \wedge \dots \wedge c_k$.

Interlude: Models of ZF

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

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

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 $s \approx t$

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 $s \in \tau\, X\, f := \exists x.\, s \approx f\, x$

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:
  - $\emptyset := \tau \perp \mathrm{elim}_\perp$

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:
  - $\emptyset := \tau\, \bot\, \mathsf{elim}_\bot$
  - $\{s, t\} := \tau\, \mathbb{B}\, (\lambda b.\, \text{if } b \text{ then } s \text{ else } t)$

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

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:
  - $\emptyset := \tau \perp \mathsf{elim}_\perp$
  - $\{s, t\} := \tau\, \mathbb{B}\, (\lambda b.\, \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t)$
  - $\omega := \tau\, \mathbb{N}\, (\lambda n.\, \overline{n})$ where $\overline{0} := \emptyset$ and $\overline{\mathsf{S}\, n} := \overline{n} \cup \{\overline{n}\}$
  - ...

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

Axioms needed in Coq:

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

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

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 $(\mathcal{T}, \approx)$ suffices (Kirst and Smolka (2019))

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:
  - $\emptyset := \tau \perp \mathsf{elim}_\perp$
  - $\{s, t\} := \tau\, \mathbb{B}\, (\lambda b.\, \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t)$
  - $\omega := \tau\, \mathbb{N}\, (\lambda n.\, \overline{n})$ where $\overline{0} := \emptyset$ and $\overline{\mathsf{S}\, n} := \overline{n} \cup \{\overline{n}\}$
  - ...

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 $(\mathcal{T}, \approx)$ suffices (Kirst and Smolka (2019))
- This yields a well-behaved model $\mathcal{S}$: quotiented, standard numbers

Sketch for ZF Set Theory

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

Sketch for ZF Set Theory

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

Reduction from PCP:

- Boolean encoding: $\overline{\mathsf{tt}} = \{\emptyset\}$ and $\overline{\mathsf{ff}} = \emptyset$
- String encoding: $\overline{\mathsf{tt}\,\mathsf{ff}\,\mathsf{ff}\,\mathsf{tt}} = (\overline{\mathsf{tt}}, (\overline{\mathsf{ff}}, (\overline{\mathsf{tt}}, (\overline{\mathsf{ff}}, \emptyset))))$
- Stack encoding: $\overline{S} = \{(\overline{s_1}, \overline{t_1}), \dots, (\overline{s_k}, \overline{t_k})\}$

Sketch for ZF Set Theory

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

Reduction from PCP:

- Boolean encoding: $\overline{\mathsf{tt}} = \{\emptyset\}$ and $\overline{\mathsf{ff}} = \emptyset$
- String encoding: $\overline{\mathsf{tt}\,\mathsf{ff}\,\mathsf{ff}\,\mathsf{tt}} = (\overline{\mathsf{tt}}, (\overline{\mathsf{ff}}, (\overline{\mathsf{tt}}, (\overline{\mathsf{ff}}, \emptyset))))$
- Stack encoding: $\overline{S} = \{(\overline{s_1}, \overline{t_1}), \dots, (\overline{s_k}, \overline{t_k})\}$
- 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 $(\emptyset, \{\_,\_\}, \bigcup, \mathcal{P}, \omega\,; \equiv, \in)$.

Reduction from PCP:

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

Sketch for ZF Set Theory

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

Reduction from PCP:

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

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

Sketch for ZF Set Theory

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

Reduction from PCP:

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

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

Theorem

PCP S *iff* ZF $\vDash \varphi_S$ *and* PCP S *iff* ZF $\vdash \varphi_S$.

Sketch for ZF Set Theory

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

Reduction from PCP:

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

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

Theorem

PCP S *iff* ZF $\vDash \varphi_S$ *and* PCP S *iff* ZF $\vdash \varphi_S$.

Proof.

Direction $\to$ by proofs in ZF and $\leftarrow$ relies on standard model $\mathcal{S}$. □

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

Aczel, P. (1978). The type theoretic interpretation of constructive set theory. In *Studies in Logic and the Foundations of Mathematics*, volume 96, pages 55–66. Elsevier.

Bauer, A. (2006). First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5 – 31. Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI).

Berger, J., Ishihara, H., and Schuster, P. (2012). The weak König lemma, Brouwer's fan theorem, De Morgan's law, and dependent choice. *Reports on Mathematical Logic*, (47):63.

Börger, E., Grädel, E., and Gurevich, Y. (1997). *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer-Verlag Berlin Heidelberg.

Coquand, T. (1986). *The calculus of constructions*. PhD thesis, INRIA.

Diener, H. (2020). Constructive Reverse Mathematics. *arXiv:1804.05495 [math]*.

Forster, Y. (2021). *Computability in Constructive Type Theory*. PhD thesis, Saarland University.

Forster, Y. (2022). Parametric church's thesis: Synthetic computability without choice. In *Logical Foundations of Computer Science: International Symposium, LFCS 2022, January 10-13, 2022*.

Forster, Y., Heiter, E., and Smolka, G. (2018). Verification of PCP-related computational reductions in Coq. In *International Conference on Interactive Theorem Proving*, pages 253–269. Springer.

Bibliography II

Forster, Y., Kirst, D., and Smolka, G. (2019). On synthetic undecidability in coq, with an application to the entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*.

Forster, Y., Kirst, D., and Wehr, D. (2020a). Completeness Theorems for First-Order Logic Analysed in Constructive Type Theory. In *Symposium on Logical Foundations Of Computer Science, 2020, Deerfield Beach, Florida, U.S.A.*

Forster, Y., Kirst, D., and Wehr, D. (2021). Completeness theorems for first-order logic analysed in constructive type theory (extended version). *Journal of Logic and Computation*, 31(1):112–151.

Forster, Y., Larchey-Wendling, D., Dudenhefner, A., Heiter, E., Kirst, D., Kunze, F., Smolka, G., Spies, S., Wehr, D., and Wuttke, M. (2020b). A Coq Library of Undecidable Problems. In *CoqPL 2020*, New Orleans, LA, United States.

Gödel, K. (1931). Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für mathematik und physik*, 38(1):173–198.

Gödel, K. (1930). Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37:349–360.

Henkin, L. (1949). The Completeness of the First-Order Functional Calculus. *The Journal of Symbolic Logic*, 14(3):159–166.

Bibliography III

Herbelin, H. and Ilik, D. (2016). An analysis of the constructive content of Henkin's proof of Gödel's completeness theorem. Draft.

Herbelin, H. and Lee, G. (2009). Forcing-based cut-elimination for Gentzen-style intuitionistic sequent calculus. In *International Workshop on Logic, Language, Information, and Computation*, pages 209–217. Springer.

Ishihara, H. (2006). Reverse Mathematics in Bishop's Constructive Mathematics. *Philosophia Scientae*, pages 43–59.

Kirst, D. and Hermes, M. (2021). Synthetic undecidability and incompleteness of first-order axiom systems in coq. In *12th International Conference on Interactive Theorem Proving (ITP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Kirst, D. and Larchey-Wendling, D. (2020). Trakhtenbrot's theorem in coq. In *International Joint Conference on Automated Reasoning*, pages 79–96. Springer.

Kirst, D. and Smolka, G. (2019). Categoricity results and large model constructions for second-order ZF in dependent type theory. *Journal of Automated Reasoning*, 63(2):415–438.

Kreisel, G. (1962). On weak completeness of intuitionistic predicate logic. *The Journal of Symbolic Logic*, 27(2):139–158.

Kreisel, G. (1965). Mathematical logic. *Lectures in modern mathematics*, 3:95–195.

Bibliography IV

Kripke, S. A. (1965). Semantical analysis of intuitionistic logic i. In *Studies in Logic and the Foundations of Mathematics*, volume 40, pages 92–130. Elsevier.

Krivtsov, V. N. (2015). Semantical completeness of first-order predicate logic and the weak fan theorem. *Studia Logica*, 103(3):623–638.

Larchey-Wendling, D. and Forster, Y. (2019). Hilbert's Tenth Problem in Coq. In *4th International Conference on Formal Structures for Computation and Deduction*, volume 131 of *LIPIcs*, pages 27:1–27:20.

Libkin, L. (2010). *Elements of Finite Model Theory*. Springer Publishing Company, Incorporated, 1st edition.

Manna, Z. (2003). *Mathematical theory of computation*. Dover Publications, Inc.

Paulin-Mohring, C. (1993). Inductive definitions in the system coq rules and properties. In *International Conference on Typed Lambda Calculi and Applications*, pages 328–345. Springer.

Post, E. L. (1946). A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52(4):264–268.

Richman, F. (1983). Church's thesis without tears. *The Journal of symbolic logic*, 48(3):797–803.

Simpson, S. G. (1985). Reverse mathematics. In *Proc. Symposia Pure Math*, volume 42, pages 461–471.

Bibliography V

Smolka, G. and Stark, K. (2016). Hereditarily Finite Sets in Constructive Type Theory. In *Interactive Theorem Proving - 7th International Conference, ITP 2016, Nancy, France, August 22-27, 2016*, volume 9807 of *LNCS*, pages 374–390. Springer.

Tarski, A. (1953). I: A general method in proofs of undecidability. In Tarski, A., editor, *Undecidable Theories*, volume 13 of *Studies in Logic and the Foundations of Mathematics*, pages 1–34. Elsevier.

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

Trakhtenbrot, B. A. (1950). Impossibility of an algorithm for the decision problem in finite classes. *Doklady Akademii Nauk SSSR*, 70(4):569–572.

Turing, A. M. (1938). On computable numbers, with an application to the entscheidungsproblem. a correction. *Proceedings of the London Mathematical Society*, 2(1):544–546.

Veldman, W. (1976). An intuitiomstic completeness theorem for intuitionistic predicate logic 1. *The Journal of Symbolic Logic*, 41(1):159–166.

Werner, B. (1997). Sets in types, types in sets. In *International Symposium on Theoretical Aspects of Computer Software*, pages 530–546. Springer.