Logika i teoria typów

Wykład 8

7 grudnia 2020

W poprzednim odcinku: logika pierwszego rzędu

Sygnatura: Ustalone symbole relacyjne, funkcyjne i stałe. Sygnatura może być nieskończona.

Termy: Zmienne $a, b, \ldots$, stałe, wyrażenia $ft_1 \ldots t_n$.

Formuły:

– atomowe: $rt_1 \ldots t_n$, $\bot$;

– $\varphi \to \psi$, $\varphi \vee \psi$, $\varphi \wedge \psi$;

– $\forall a\, \varphi$, $\exists a\, \varphi$.

Zmienne wolne: $\mathrm{FV}(\forall a\, \varphi) = \mathrm{FV}(\exists a\, \varphi) = \mathrm{FV}(\varphi) - \{a\}$.

Utożsamiamy formuły różniące się tylko zmiennymi związanymi.

Kripke semantics for first-order logic

Model: $\mathcal{C} = \langle C, \leq, \{\mathcal{A}_c \mid c \in C\}\rangle$

Kripke semantics for first-order logic

Model: $\mathcal{C} = \langle C, \leq, \{\mathcal{A}_c \mid c \in C\}\rangle$

Dla $c \leq d \in C$ struktura $\mathcal{A}_c$ zawiera się w $\mathcal{A}_d$, tj.

– nośnik $|\mathcal{A}_c|$ jest podzbiorem $|\mathcal{A}_d|$;

– relacje w $\mathcal{A}_c$ są podzbiorami relacji w $\mathcal{A}_d$;

– funkcje w $\mathcal{A}_c$ są obcięciami funkcji w $\mathcal{A}_d$.

Semantyka Kripkego

$c, \varrho \Vdash rt_1 \dots t_n \equiv \mathcal{A}_c, \varrho \models rt_1 \dots t_n$ (klasycznie);

$c, \varrho \nVdash \bot$;

$c, \varrho \Vdash \varphi \vee \psi \equiv c, \varrho \Vdash \varphi$ lub $c, \varrho \Vdash \psi$;

$c, \varrho \Vdash \varphi \wedge \psi \equiv c, \varrho \Vdash \varphi$ oraz $c, \varrho \Vdash \psi$;

$c, \varrho \Vdash \varphi \to \psi \equiv$ jeśli $c \leq c'$ i $c', \varrho \Vdash \varphi$, to $c', \varrho \Vdash \psi$;

$c, \varrho \Vdash \exists a \varphi \equiv c, \varrho[a \mapsto \mathbf{a}] \Vdash \varphi$, dla pewnego $\mathbf{a} \in \mathcal{A}_c$;

$c, \varrho \Vdash \forall a \varphi \equiv$ jeśli $c \leq c'$ i $\mathbf{a} \in \mathcal{A}_{c'}$, to $c', \varrho[a \mapsto \mathbf{a}] \Vdash \varphi$.

Twierdzenie o poprawności

Lemat 1: Jeśli $c, \varrho \Vdash \varphi$ i $c \leq c'$, to $c', \varrho \Vdash \varphi$.

Twierdzenie o poprawności

Lemat 1: Jeśli $c, \varrho \Vdash \varphi$ i $c \leq c'$, to $c', \varrho \Vdash \varphi$.

Lemat 2: $c, \varrho \Vdash \varphi[a := t] \quad \Leftrightarrow \quad c, \varrho[a \mapsto [\![t]\!]_\varrho] \Vdash \varphi$.

Twierdzenie o poprawności

Lemat 1: Jeśli $c, \varrho \Vdash \varphi$ i $c \leq c'$, to $c', \varrho \Vdash \varphi$.

Lemat 2: $c, \varrho \Vdash \varphi[a := t] \quad \Leftrightarrow \quad c, \varrho[a \mapsto [\![t]\!]_\varrho] \Vdash \varphi$.

Twierdzenie: *Jeśli* $\Gamma \vdash \varphi$ *oraz* $c, \varrho \Vdash \Gamma$, *to* $c, \varrho \Vdash \varphi$.

(W skrócie: jeśli $\Gamma \vdash \varphi$, *to* $\Gamma \Vdash \varphi$.*)*

Dowód: Indukcja.

$\circ$

Przykład: $\not\Vdash \neg\neg\forall a\,(ra \vee \neg ra)$

Model $\mathcal{C} = \langle \mathbb{N}, \leq, \{\mathcal{A}_n \mid n \in \mathbb{N}\}\rangle$, porządek jak zwykle.
Struktury $\mathcal{A}_n = \langle \mathbb{N}, r^n\rangle$, gdzie $r^n = \{m \mid m < n\}$.

Przykład: $\not\Vdash \neg\neg\forall a\,(ra \vee \neg ra)$

Model $\mathcal{C} = \langle \mathbb{N}, \leq, \{\mathcal{A}_n \mid n \in \mathbb{N}\}\rangle$, porządek jak zwykle.
Struktury $\mathcal{A}_n = \langle \mathbb{N}, r^n\rangle$, gdzie $r^n = \{m \mid m < n\}$.

W tym modelu żaden stan nie wymusza $\forall a\,(ra \vee \neg ra)$.
Zatem każdy stan wymusza $\neg\forall a\,(ra \vee \neg ra)$.

Przykład: $\not\Vdash \neg\neg\forall a\,(ra \vee \neg ra)$

Model $\mathcal{C} = \langle \mathbb{N}, \leq, \{\mathcal{A}_n \mid n \in \mathbb{N}\}\rangle$, porządek jak zwykle.
Struktury $\mathcal{A}_n = \langle \mathbb{N}, r^n\rangle$, gdzie $r^n = \{m \mid m < n\}$.

W tym modelu żaden stan nie wymusza $\forall a\,(ra \vee \neg ra)$.
Zatem każdy stan wymusza $\neg\forall a\,(ra \vee \neg ra)$.

Uwaga: (1) Nie działa twierdzenie Gliwienki.
(2) Spełnialność intuicjonistyczna nie implikuje klasycznej.

Przykład: $\not\Vdash \neg\neg\forall a\,(ra \vee \neg ra)$

Model $\mathcal{C} = \langle \mathbb{N}, \leq, \{\mathcal{A}_n \mid n \in \mathbb{N}\}\rangle$, porządek jak zwykle.
Struktury $\mathcal{A}_n = \langle \mathbb{N}, r^n\rangle$, gdzie $r^n = \{m \mid m < n\}$.

W tym modelu żaden stan nie wymusza $\forall a\,(ra \vee \neg ra)$.
Zatem każdy stan wymusza $\neg\forall a\,(ra \vee \neg ra)$.

Uwaga: (1) Nie działa twierdzenie Gliwienki.
(2) Spełnialność intuicjonistyczna nie implikuje klasycznej.

Fakt: *Jeśli model $\mathcal{C}$ ma skończony zbiór stanów,*
to $\mathcal{C} \Vdash \neg\neg\forall a\,(ra \vee \neg ra)$.

Przykład: $\nvdash \forall x(p \vee r(x)) \to p \vee \forall x.r(x)$

Model $\mathcal{C} = \langle \{1,2\}, \leq, \{\mathcal{A}_1, \mathcal{A}_2\} \rangle$, gdzie $1 < 2$, oraz

$$\mathcal{A}_1 = \langle \{1\}, r^1, p^1 \rangle, \qquad \mathcal{A}_2 = \langle \{1,2\}, r^2, p^2 \rangle$$
$$r^1 = \{1\}, p^1 = \bot, \qquad r^2 = \{1\}, p^2 = \top$$

(Relacje $\bot$ i $\top$ są zeroargumentowe.)

Przykład: $\nVdash \forall x(p \vee r(x)) \to p \vee \forall x.r(x)$

Model $\mathcal{C} = \langle \{1,2\}, \leq, \{\mathcal{A}_1, \mathcal{A}_2\} \rangle$, gdzie $1 < 2$, oraz

$$\mathcal{A}_1 = \langle \{1\}, r^1, p^1 \rangle, \qquad \mathcal{A}_2 = \langle \{1,2\}, r^2, p^2 \rangle$$
$$r^1 = \{1\}, p^1 = \bot, \qquad r^2 = \{1\}, p^2 = \top$$

(Relacje $\bot$ i $\top$ są zeroargumentowe.)

W tym modelu $1 \Vdash \forall x(p \vee r(x))$, ale $1 \nVdash p \vee \forall x.r(x)$.

Przykład: $\not\Vdash \forall x(p \vee r(x)) \to p \vee \forall x.r(x)$

Model $\mathcal{C} = \langle \{1,2\}, \leq, \{\mathcal{A}_1, \mathcal{A}_2\} \rangle$, gdzie $1 < 2$, oraz

$$\mathcal{A}_1 = \langle \{1\}, r^1, p^1 \rangle, \qquad \mathcal{A}_2 = \langle \{1,2\}, r^2, p^2 \rangle$$
$$r^1 = \{1\}, p^1 = \bot, \qquad r^2 = \{1\}, p^2 = \top$$

(Relacje $\bot$ i $\top$ są zeroargumentowe.)

W tym modelu $1 \Vdash \forall x(p \vee r(x))$, ale $1 \not\Vdash p \vee \forall x.r(x)$.

Fakt (Schemat Grzegorczyka)

Jeśli w modelu $\mathcal{C}$ wszystkie $|A_c|$ są takie same, to

$$\mathcal{C} \Vdash \forall a(\psi \vee \varphi(a)) \to \psi \vee \forall a\, \varphi(a).$$

First-order games (function-free)

If ∃ros chooses an assumption α in position $\Gamma \vdash \tau$

a1) If α is an assumption $\beta \to \gamma$ then ∀phrodite chooses between positions $\Gamma, \gamma \vdash \tau$ and $\Gamma \vdash \beta$.

a2) If α is an assumption $\beta \vee \gamma$ then ∀phrodite chooses between positions $\Gamma, \beta \vdash \tau$ and $\Gamma, \gamma \vdash \tau$.

a3) If α is an assumption $\beta \wedge \gamma$ then ∀phrodite has no choice and the next position is $\Gamma, \beta, \gamma \vdash \tau$.

a4) If α is an assumption $\forall x\, \varphi$, then ∃ros also chooses an arbitrary variable y. Then the next position is $\Gamma, \varphi[x := y] \vdash \tau$.

a5) If α is an assumption $\exists x\, \varphi$ then the next position is $\Gamma, \varphi[x := z] \vdash \tau$, where z is fresh.

If ∃ros chooses an aim α in position $\Gamma \vdash \tau$

Now $\tau = \alpha \vee \beta$ or $\tau = \alpha$.

b1) If α is an aim $\beta \to \gamma$ then the next position is $\Gamma, \beta \vdash \gamma$.

b2) If α is an aim $\beta \wedge \gamma$ then ∀phrodite a chooses between positions $\Gamma \vdash \beta$ and $\Gamma \vdash \gamma$.

b3) If the aim α is an atom or a disjunction then the next position is $\Gamma \vdash \alpha$.

b4) If α is an aim $\forall x\, \varphi$ then the next position is $\Gamma \vdash \varphi[x := z]$, where z is fresh.

b5) If α is an aim $\exists x\, \varphi$ then ∃ros chooses an arbitrary variable y and the next position is $\Gamma \vdash \varphi[x := y]$.

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

∃ : `intro !`

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q$?

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \vdash Q$?

$\exists$: `apply` X !

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q$?

$\exists$: `apply` X !

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; \forall x\,(P(x) \vee (P(x) \to Q))$?

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q$?

$\exists$: `apply` X !

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; \forall x\,(P(x) \vee (P(x) \to Q))$?

$\exists$: `intro !`

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q$?

$\exists$: `apply` X !

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; \forall x\,(P(x) \vee (P(x) \to Q))$?

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; P(x_0) \vee (P(x_0) \to Q)$?

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q$?

$\exists$: `apply` X !

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; \forall x\,(P(x) \vee (P(x) \to Q))$?

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; P(x_0) \vee (P(x_0) \to Q)$?

$\exists$: `right; intro !`

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q$?

$\exists$: `apply` X !

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; \forall x\,(P(x) \vee (P(x) \to Q))$?

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; P(x_0) \vee (P(x_0) \to Q)$?

$\exists$: `right; intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q,\; Y_0 : P(x_0) \;\vdash\; Q$?

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q$?

$\exists$: `apply` X !

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; \forall x\,(P(x) \vee (P(x) \to Q))$?

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; P(x_0) \vee (P(x_0) \to Q)$?

$\exists$: `right; intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q,\; Y_0 : P(x_0) \;\vdash\; Q$?

$(\ldots)$

Example: $(\forall x\,(P(x) \vee (P(x) \to Q)) \to Q) \to Q$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; Q\;?$

$\exists$: `apply` X `!`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; \forall x\,(P(x) \vee (P(x) \to Q))\;?$

$\exists$: `intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q \;\vdash\; P(x_0) \vee (P(x_0) \to Q)\,?$

$\exists$: `right; intro !`

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q,\; Y_0 : P(x_0) \;\vdash\; Q\;?$

$(\ldots)$

$\forall$: $X : \forall x\,(P(x) \vee (P(x) \to Q)) \to Q,\; Y_0 : P(x_0),\; Y_1 : P(x_1) \;\vdash\; Q\,?$

Strategies of the players

Łatwe: *Każda strategia ∃rosa określa dowód normalny.*

Trudne: Czy strategia ∀frodyty definiuje model?

Przykład: Ta formuła wymaga modelu o nieskończonych dziedzinach:

$$\forall x \exists y\, R(x,y) \to \forall xyz\, (R(x,y) \to R(y,z) \to R(x,z)) \to \exists x\, R(x,x)$$

Ale pozycje są skończone!

Stany modelu to muszą być rozgrywki (ciągi pozycji).

Is any strategy a counter-model?

States in a model must be infinite plays $\mathfrak{p}$, where the goal $\tau_{\mathfrak{p}}$ is the same in almost all positions and the assumptions sum up to a set $\Gamma_{\mathfrak{p}}$.

Rozszerzanie rozgrywek: $\mathfrak{p}_1 \leq \mathfrak{p}_2$ musi oznaczać,
że każdy krok $\mathfrak{p}_1$ ma odpowiednik w $\mathfrak{p}_2$.

We need the following lemma to make this work:

Every state $\mathfrak{p}$ forces all formulas in $\Gamma_{\mathfrak{p}}$,
and does not force the formula $\tau_{\mathfrak{p}}$.

Is any strategy a counter-model?

Not necessarily. *La donna è mobile:* ∀phrodite's strategy may be non-uniform but still winning.

Example: $\Phi, \Psi, \Theta \vdash P \to Q$, where:

$\Phi = \forall x(L(x) \vee R(x))$;

$\Psi = \forall x\,(L(x) \wedge R(x) \to Q)$;

$\Theta = \forall x \exists y\, S(x, y)$.

As long as ∃ros refers only to assumptions, using variables x_i in Φ, ∀phrodite chooses $L(x_i)$.

When he finally plays the aim, she changes her mind and chooses $R(x_i)$ for the new variables.

There is a play where ∃ros never addresses the aim. It cannot be reasonably "extended" to a play where P is forced. That play wrongly forces $P \to Q$, the lemma fails.

Counter-model out of the full strategy

Full strategy: consists of all "eventually winning" positions of ∀phrodite. It is a "nondeterministic" strategy including all safe moves of ∀phrodite.

Model: States of the model are *saturated plays*: those where every "safe static turn" is actually played.

Grzegorczyk scheme

$$\forall x(P \vee R(x)) \vdash P \vee \forall x.R(x)$$

Phase I: If $\exists$ros chooses $P \vee R(x')$, for some x', then $\forall$phrodite always responds by adding assumption $R(x')$

Grzegorczyk scheme

$$\forall x(P \vee R(x)) \vdash P \vee \forall x.R(x)$$

Phase I: If $\exists$ros chooses $P \vee R(x')$, for some x', then $\forall$phrodite always responds by adding assumption $R(x')$

Phase II: Eventually $\exists$ros may address an aim at the rhs.

If he chooses P, then $\forall$phrodite plays as in Phase I and wins.

Otherwise $R(y)$ is the new goal (with fresh y).

Grzegorczyk scheme

$$\forall x(P \vee R(x)) \vdash P \vee \forall x.R(x)$$

Phase I: If $\exists$ros chooses $P \vee R(x')$, for some x', then
$\forall$phrodite always responds by adding assumption $R(x')$

Phase II: Eventually $\exists$ros may address an aim at the rhs.

If he chooses P, then $\forall$phrodite plays as in Phase I and wins.

Otherwise $R(y)$ is the new goal (with fresh y).

Phase III: If now $\exists$ros chooses $P \vee R(y)$,
then $\forall$phrodite can safely add P to assumptions.
(There is no more P at rhs.)

Grzegorczyk scheme: a counter-model

Saturated plays in the game of $\forall x(P \vee R(x)) \vdash P \vee \forall x.R(x)$ are of three kinds, depending on the (eventual) rhs:

1. $R(x'), \ldots \quad \vdash P \vee \forall x.R(x)$; $\{x', \ldots\}$
2. $R(x'), \ldots \quad \vdash P$; $\{x', \ldots\}$
3. $R(x'), \ldots, P \vdash R(y)$. $\{y, x', \ldots\}$

States of type 1 can be "extended" to type 2 and 3.

$\circ$

Wnioski z determinacji dla gry pierwszego rzędu

Twierdzenie: Albo istnieje dowód normalny, albo kontrmodel.

Wnioski:

(1) Pełność: jeśli $\Vdash \varphi$, to $\vdash \varphi$;

(2) Semantyczna normalizacja:
jeśli formuła φ ma dowód, to ma dowód normalny.

$\circ$

Postaci normalne

Lemat: *Let M be a term in normal form without free proof variables. Then:*

0) *M is not of type $\bot$.*

Postaci normalne

Lemat: *Let M be a term in normal form without free proof variables. Then:*

0) *M is not of type $\bot$.*
1) *If $M : \varphi \vee \psi$ then $M = \mathbf{in}_i\, N$, for some i and N.*

Postaci normalne

Lemat: *Let M be a term in normal form without free proof variables. Then:*

0) *M is not of type $\bot$.*
1) *If $M : \varphi \vee \psi$ then $M = \mathbf{in}_i\, N$, for some i and N.*
2) *If $M : \exists a \varphi$ then $M = [N, t]$ for some N, t.*

Postaci normalne

Lemat: *Let M be a term in normal form without free proof variables. Then:*

0) *M is not of type $\bot$.*
1) *If $M : \varphi \vee \psi$ then $M = \mathbf{in}_i\, N$, for some i and N.*
2) *If $M : \exists a \varphi$ then $M = [N, t]$ for some N, t.*

Wnioski:

- *Logika pierwszego rzędu jest niesprzeczna ($\nvdash \bot$).*

Postaci normalne

Lemat: *Let M be a term in normal form without free proof variables. Then:*

0) *M is not of type $\bot$.*
1) *If $M : \varphi \vee \psi$ then $M = \mathbf{in}_i\, N$, for some i and N.*
2) *If $M : \exists a \varphi$ then $M = [N, t]$ for some N, t.*

Wnioski:

- *Logika pierwszego rzędu jest niesprzeczna ($\nvdash \bot$).*
- *Jeśli $\vdash \varphi \vee \psi$, to $\vdash \varphi$ lub $\vdash \psi$.*

Postaci normalne

Lemat: *Let M be a term in normal form without free proof variables. Then:*

0) *M is not of type $\bot$.*
1) *If $M : \varphi \vee \psi$ then $M = \mathbf{in}_i\, N$, for some i and N.*
2) *If $M : \exists a \varphi$ then $M = [N, t]$ for some N, t.*

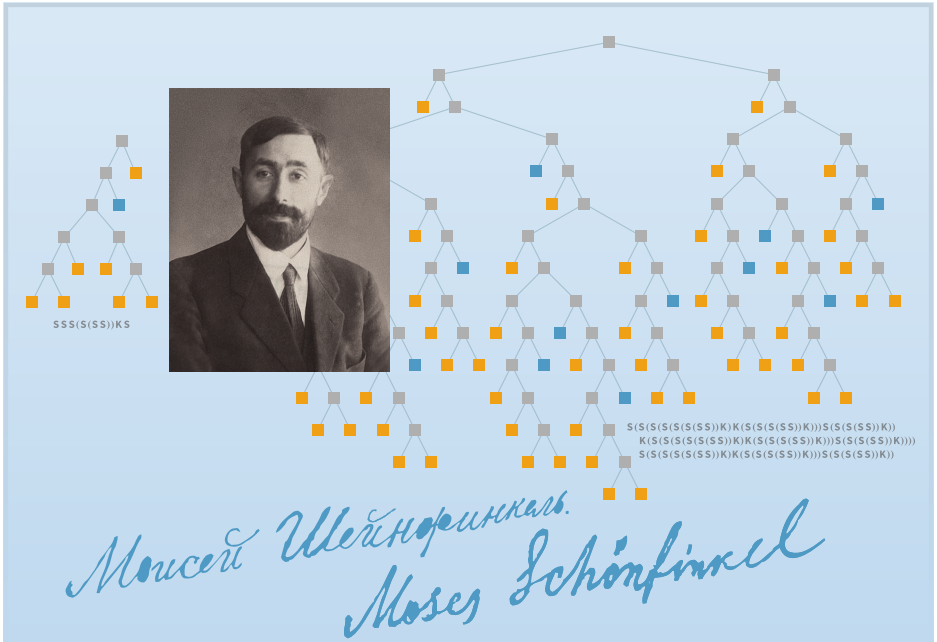
Wnioski:

- *Logika pierwszego rzędu jest niesprzeczna ($\nvdash \bot$).*
- *Jeśli $\vdash \varphi \vee \psi$, to $\vdash \varphi$ lub $\vdash \psi$.*
- *Jeśli $\vdash \exists a\, \varphi$, to $\vdash \varphi[a := t]$.*

Przerwa na reklamę

Sto lat logiki kombinatorycznej.

Sto lat logiki kombinatorycznej. Dzisiaj.



Combinators: A 100-Year Celebration

Livestreamed Event

On December 7, 1920, at 6pm German time, Moses Schönfinkel gave the talk that introduced combinators. We are celebrating its 100th anniversary with a public livestream about the history and future of combinators, and their significance for computation.

December 7, 2020, at 6pm CET/12 noon EST/9am PST

JOIN LIVE EITHER OF:

youtube.com/user/WolframResearch

twitch.tv/Stephen_Wolfram

Erasing dependencies

Wycieranie zależności z typów

Wycieranie zależności z typów

- $\kappa(Pt_1 \ldots t_n) = P$ (zmienna zdaniowa);

Wycieranie zależności z typów

- $\kappa(Pt_1 \ldots t_n) = P$ (zmienna zdaniowa);
- $\kappa(\varphi \to \psi) = \kappa(\varphi) \to \kappa(\psi)$;

Wycieranie zależności z typów

- $\kappa(Pt_1 \ldots t_n) = P$ (zmienna zdaniowa);
- $\kappa(\varphi \to \psi) = \kappa(\varphi) \to \kappa(\psi)$;
- $\kappa(\varphi \vee \psi) = \kappa(\varphi) \vee \kappa(\psi)$;

Wycieranie zależności z typów

- $\kappa(Pt_1 \ldots t_n) = P$ (zmienna zdaniowa);
- $\kappa(\varphi \to \psi) = \kappa(\varphi) \to \kappa(\psi)$;
- $\kappa(\varphi \vee \psi) = \kappa(\varphi) \vee \kappa(\psi)$;
- $\kappa(\varphi \wedge \psi) = \kappa(\varphi) \wedge \kappa(\psi)$;

Wycieranie zależności z typów

- $\kappa(Pt_1 \ldots t_n) = P$ (zmienna zdaniowa);
- $\kappa(\varphi \to \psi) = \kappa(\varphi) \to \kappa(\psi)$;
- $\kappa(\varphi \vee \psi) = \kappa(\varphi) \vee \kappa(\psi)$;
- $\kappa(\varphi \wedge \psi) = \kappa(\varphi) \wedge \kappa(\psi)$;
- $\kappa(\bot) = \bot$;

Wycieranie zależności z typów

- $\kappa(Pt_1 \ldots t_n) = P$ (zmienna zdaniowa);
- $\kappa(\varphi \to \psi) = \kappa(\varphi) \to \kappa(\psi)$;
- $\kappa(\varphi \vee \psi) = \kappa(\varphi) \vee \kappa(\psi)$;
- $\kappa(\varphi \wedge \psi) = \kappa(\varphi) \wedge \kappa(\psi)$;
- $\kappa(\bot) = \bot$;
- $\kappa(\exists a \varphi) = \kappa(\varphi)$;

Wycieranie zależności z typów

- $\kappa(Pt_1 \dots t_n) = P$ (zmienna zdaniowa);
- $\kappa(\varphi \to \psi) = \kappa(\varphi) \to \kappa(\psi)$;
- $\kappa(\varphi \vee \psi) = \kappa(\varphi) \vee \kappa(\psi)$;
- $\kappa(\varphi \wedge \psi) = \kappa(\varphi) \wedge \kappa(\psi)$;
- $\kappa(\bot) = \bot$;
- $\kappa(\exists a \varphi) = \kappa(\varphi)$;
- $\kappa(\forall a \varphi) = \kappa(\varphi)$.

Przykład

$$\kappa(\forall x.\, (P \vee Q(x)) \to P \vee \forall x.\, Q(x)) = P \vee Q \to P \vee Q$$

$\circ$

Wycieranie zależności z termów (1)

- $\kappa(x^{\varphi}) = x^{\kappa(\varphi)}$
- $\kappa(\lambda x{:}\varphi.M) = \lambda x{:}\kappa(\varphi).\kappa(M)$
- $\kappa(MN) = \kappa(M)\kappa(N)$
- $\kappa(\langle M, N\rangle) = \langle \kappa(M), \kappa(N)\rangle$
- $\kappa(\pi_i(M)) = \pi_i(\kappa(M))$
- $\kappa(\mathbf{in}_1\, M) = \mathbf{in}_1\, \kappa(M)$
- $\kappa(\mathbf{in}_2\, M) = \mathbf{in}_2\, \kappa(M)$
- $\kappa(\mathbf{case}\, M\, \mathbf{of}\, [x]P, [y]Q) = \mathbf{case}\, \kappa(M)\, \mathbf{of}\, [x]\kappa(P), [y]\kappa(Q)$

Wycieranie zależności z termów (2)

- $\kappa(\lambda a.M) = \kappa(M)$.
- $\kappa(Mt) = \kappa(M)$.
- $\kappa([t, M]) = \kappa(M)$.
- $\kappa(\textbf{unpack } M \textbf{ as } [a, y{:}\varphi] \textbf{ in } N) = (\lambda y{:}\kappa(\varphi).\kappa(N))\kappa(M)$.

Wycieranie zależności z termów (2)

- $\kappa(\lambda a.M) = \kappa(M)$.
- $\kappa(Mt) = \kappa(M)$.
- $\kappa([t, M]) = \kappa(M)$.
- $\kappa(\textbf{unpack } M \textbf{ as } [a, y{:}\varphi] \textbf{ in } N) = (\lambda y{:}\kappa(\varphi).\kappa(N))\kappa(M)$.

Lemat: *Jeśli* $\Gamma \vdash M : \varphi$, *to* $\kappa(\Gamma) \vdash \kappa(M) : \kappa(\varphi)$.

$\circ$

Example use: strong normalization

Lemat: *Jeśli* $M \to_\beta M'$, *to* $\kappa(M) \twoheadrightarrow^+ \kappa(M')$, *z wyjątkiem redukcji „indywiduowych” (gdy redeks* $(\lambda a.N)t$ *zamieniamy na* $N[a := t]$*). W tym przypadku* $\kappa(M) = \kappa(M')$.

Example use: strong normalization

Lemat: *Jeśli* $M \to_\beta M'$, *to* $\kappa(M) \twoheadrightarrow^+ \kappa(M')$, *z wyjątkiem redukcji „indywiduowych” (gdy redeks* $(\lambda a.N)t$ *zamieniamy na* $N[a := t]$*). W tym przypadku* $\kappa(M) = \kappa(M')$.

Na przykład:

$$\kappa(\textbf{unpack } [t, M] \textbf{ as } [a, y{:}\varphi] \textbf{ in } N) = (\lambda y.\, \kappa(N))\kappa(M)$$
$$\to \kappa(N)[y := \kappa(M)] = \kappa(N[a := t][y := M))$$

Example use: strong normalization

Lemat: *Jeśli* $M \to_\beta M'$, *to* $\kappa(M) \twoheadrightarrow^+ \kappa(M')$, *z wyjątkiem redukcji „indywiduowych” (gdy redeks* $(\lambda a.N)t$ *zamieniamy na* $N[a := t]$*). W tym przypadku* $\kappa(M) = \kappa(M')$.

Na przykład:

$$\kappa(\textbf{unpack } [t, M] \textbf{ as } [a, y{:}\varphi] \textbf{ in } N) = (\lambda y.\, \kappa(N))\kappa(M)$$
$$\to \kappa(N)[y := \kappa(M)] = \kappa(N[a := t][y := M))$$

Lemat: *Każdy ciąg redukcji indywiduowych jest skończony.*

Dowód: Maleje liczba lambd.

Silna normalizacja

Twierdzenie: *Termy dla logiki pierwszego rzędu mają własność silnej normalizacji ze względu na beta-redukcje.*

Silna normalizacja

Twierdzenie: *Termy dla logiki pierwszego rzędu mają własność silnej normalizacji ze względu na beta-redukcje.*

Dowód: Jeśli $M_1 \to M_2 \to \cdots$ to $\kappa(M_1) \twoheadrightarrow \kappa(M_2) \twoheadrightarrow \cdots$

Nieskończenie wiele razy zachodzi „prawdziwa” redukcja.

Silna normalizacja

Twierdzenie: *Termy dla logiki pierwszego rzędu mają własność silnej normalizacji ze względu na beta-redukcje.*

Dowód: Jeśli $M_1 \to M_2 \to \cdots$ to $\kappa(M_1) \twoheadrightarrow \kappa(M_2) \twoheadrightarrow \cdots$

Nieskończenie wiele razy zachodzi „prawdziwa" redukcja.

Silna normalizacja dla permutacji też zachodzi.
Dowód podobny jak dla rachunku zdań.

Erasing dependencies from Peano Arithmetic

Weźmy aksjomat indukcji Peana z 1889 roku:

$$\tau(0) \to \forall y(\mathbf{int}(y) \to \tau(y) \to \tau(\mathsf{s}y)) \to \forall x(\mathbf{int}(x) \to \tau(x)).$$

Erasing dependencies from Peano Arithmetic

Weźmy aksjomat indukcji Peana z 1889 roku:

$$\tau(0) \to \forall y(\mathbf{int}(y) \to \tau(y) \to \tau(\mathsf{s}y)) \to \forall x(\mathbf{int}(x) \to \tau(x)).$$

Napiszmy go trochę inaczej:

Erasing dependencies from Peano Arithmetic

Weźmy aksjomat indukcji Peana z 1889 roku:

$$\tau(0)\to\forall y(\mathbf{int}(y)\to\tau(y)\to\tau(\mathsf{s}y))\to\forall x(\mathbf{int}(x)\to\tau(x)).$$

Napiszmy go trochę inaczej:

$$\forall x(\mathbf{int}(x)\to\tau(0)\to\forall y(\mathbf{int}(y)\to\tau(y)\to\tau(\mathsf{s}y))\to\tau(x)).$$

Erasing dependencies from Peano Arithmetic

Weźmy aksjomat indukcji Peana z 1889 roku:

$$\tau(0) \to \forall y(\mathbf{int}(y) \to \tau(y) \to \tau(\mathsf{s}y)) \to \forall x(\mathbf{int}(x) \to \tau(x)).$$

Napiszmy go trochę inaczej:

$$\forall x(\mathbf{int}(x) \to \tau(0) \to \forall y(\mathbf{int}(y) \to \tau(y) \to \tau(\mathsf{s}y)) \to \tau(x)).$$

I wytrzyjmy zależności:

Erasing dependencies from Peano Arithmetic

Weźmy aksjomat indukcji Peana z 1889 roku:

$$\tau(0)\to\forall y(\mathbf{int}(y)\to\tau(y)\to\tau(\mathsf{s}y))\to\forall x(\mathbf{int}(x)\to\tau(x)).$$

Napiszmy go trochę inaczej:

$$\forall x(\mathbf{int}(x)\to\tau(0)\to\forall y(\mathbf{int}(y)\to\tau(y)\to\tau(\mathsf{s}y))\to\tau(x)).$$

I wytrzyjmy zależności:

$$\mathbf{int}\to\tau\to(\mathbf{int}\to\tau\to\tau)\to\tau.$$

System T Gödla

Typy: Typy proste zbudowane z jednej stałej typowej **int**.
(Czasem dodaje się **bool**, produkty itp.)

Termy: Jak w rachunku lambda z typami prostymi plus stałe:

$$0 : \mathbf{int} \qquad\qquad \mathsf{s} : \mathbf{int} \to \mathbf{int}$$

$$\mathsf{R}_\tau : \mathbf{int} \to \tau \to (\mathbf{int} \to \tau \to \tau) \to \tau,$$

System T Gödla

Typy: Typy proste zbudowane z jednej stałej typowej **int**.
(Czasem dodaje się **bool**, produkty itp.)

Termy: Jak w rachunku lambda z typami prostymi plus stałe:

$$0 : \mathbf{int} \qquad \mathsf{s} : \mathbf{int} \to \mathbf{int}$$

$$\mathbf{R}_\tau : \mathbf{int} \to \tau \to (\mathbf{int} \to \tau \to \tau) \to \tau,$$

Redukcja: Zwykła beta-redukcja oraz:

$$\mathbf{R}_\tau\, 0\, P\, Q \Rightarrow P \qquad \mathbf{R}_\tau(\mathsf{s}\, n)\, P\, Q \Rightarrow Q\, n\, (\mathbf{R}_\tau\, n\, P\, Q)$$

Przykład: Funkcja poprzednika

$$\mathit{pred} = \lambda n^{\mathbf{int}}.\, \mathbf{R}_{\mathbf{int}}\, n\, 0\, (\lambda xy.\, x)$$

Twierdzenie Kreisela

Jeśli $\quad \mathrm{PA} \vdash \forall a(\mathbf{int}(a) \to \exists b\,(\mathbf{int}(b) \wedge W(a,b)))$,

gdzie W jest pierwotnie rekurencyjne.

to istnieje dowód konstruktywny (w arytmetyce Heytinga).

$\circ$

Program extraction

Suppose we can prove in HA/PA:
$\forall a(\mathbf{int}(a) \to \exists b\,(\mathbf{int}(b) \wedge W(a,b)))$.

Program extraction

Suppose we can prove in HA/PA:
$\forall a(\mathbf{int}(a) \to \exists b\,(\mathbf{int}(b) \wedge W(a,b)))$.

This theorem erases to a type $\mathbf{int} \to \mathbf{int} \times \tau$

Program extraction

Suppose we can prove in HA/PA:
$\forall a(\mathbf{int}(a) \to \exists b\,(\mathbf{int}(b) \wedge W(a,b)))$.

This theorem erases to a type $\mathbf{int} \to \mathbf{int} \times \tau$
The proof erases to a term $F : \mathbf{int} \to \mathbf{int} \times \tau$.

Program extraction

Suppose we can prove in HA/PA:
$\forall a(\mathbf{int}(a) \to \exists b\,(\mathbf{int}(b) \wedge W(a,b)))$.

This theorem erases to a type $\mathbf{int} \to \mathbf{int} \times \tau$
The proof erases to a term $F : \mathbf{int} \to \mathbf{int} \times \tau$.

The term $\lambda a.\pi_1(Fa) : \mathbf{int} \to \mathbf{int}$ defines a function f such that:

$$\forall n \in \mathbb{N}.\, W(n, f(n))).$$

Własności systemu $\mathbf{T}$

- System $\mathbf{T}$ ma własność silnej normalizacji (dowód „metodą Taita").
- Definiowalne są wszystkie funkcje dowodliwie rekurencyjne w PA.
- Własność SN dla systemu $\mathbf{T}$ jest niezależna od PA.

o

Metoda Taita

Definition: *Stable (computable,reducible...)* terms:

- $[\![\mathbf{int}]\!] := \mathrm{SN}$;
- $[\![\tau \to \sigma]\!] := \{M \mid \forall N(N \in [\![\tau]\!] \Rightarrow MN \in [\![\sigma]\!])\}$;

Główny lemat:

- Termy stabilne mają własność SN.
- Każdy term jest stabilny.

To jest trudne

Metoda Taita (zwana też metodą obliczalności) nie formalizuje się w PA, bo posługuje się dowolnymi zbiorami termów. Tego się nie da zakodować liczbami.

To jest trudne

Metoda Taita (zwana też metodą obliczalności) nie formalizuje się w PA, bo posługuje się dowolnymi zbiorami termów. Tego się nie da zakodować liczbami.

Strong normalization for system T cannot be derived in PA.

Why?

Because it implies consistency of PA (there is no proof of $\bot$) and that cannot be proven within PA.

Logika drugiego rzędu,

czyli Polimorfizm

Klasyczna logika drugiego rzędu

Składnia: Zmienne relacyjne i kwantyfikatory $\forall R, \exists R$.

Klasyczna logika drugiego rzędu

Składnia: Zmienne relacyjne i kwantyfikatory $\forall R, \exists R$.

Semantyka w stylu Tarskiego: Interpretujemy zmienne relacyjne jako relacje. Na przykład formuła

$$Nat(a) = \forall R(\forall b(R(b) \to R(s\,b)) \to R(0) \to R(a))$$

definiuje standardowe liczby naturalne.

Klasyczna logika drugiego rzędu

Składnia: Zmienne relacyjne i kwantyfikatory $\forall R, \exists R$.

Semantyka w stylu Tarskiego: Interpretujemy zmienne relacyjne jako relacje. Na przykład formuła

$$Nat(a) = \forall R(\forall b(R(b) \to R(s\,b)) \to R(0) \to R(a))$$

definiuje standardowe liczby naturalne.

Wniosek: *Aksjomaty Peana plus* $\forall a Nat(a)$ *definiują standardowy model arytmetyki z dokładnością do izomorfizmu.*

Klasyczna logika drugiego rzędu

Składnia: Zmienne relacyjne i kwantyfikatory $\forall R, \exists R$.

Semantyka w stylu Tarskiego: Interpretujemy zmienne relacyjne jako relacje. Na przykład formuła

$$Nat(a) = \forall R(\forall b(R(b) \to R(sb)) \to R(0) \to R(a))$$

definiuje standardowe liczby naturalne.

Wniosek: *Aksjomaty Peana plus* $\forall a Nat(a)$ *definiują standardowy model arytmetyki z dokładnością do izomorfizmu.*

Wniosek: *Zbiór tautologii drugiego rzędu nie jest rekurencyjnie przeliczalny (bo* $\mathrm{Th}(\mathbb{N})$ *nie jest).*

Klasyczna logika drugiego rzędu

Składnia: Zmienne relacyjne i kwantyfikatory $\forall R, \exists R$.

Semantyka w stylu Tarskiego: Interpretujemy zmienne relacyjne jako relacje. Na przykład formuła

$$Nat(a) = \forall R(\forall b(R(b) \to R(sb)) \to R(0) \to R(a))$$

definiuje standardowe liczby naturalne.

Wniosek: *Aksjomaty Peana plus* $\forall a Nat(a)$ *definiują standardowy model arytmetyki z dokładnością do izomorfizmu.*

Wniosek: *Zbiór tautologii drugiego rzędu nie jest rekurencyjnie przeliczalny (bo* $\mathrm{Th}(\mathbb{N})$ *nie jest).*

Wniosek: *Nie ma pełnego systemu wnioskowania.*

Siła wyrazu języka drugiego rzędu

Przykład: Dodawanie liczb naturalnych ($m + n = k$):

$$\forall R(\forall a(Ra0a) \to \forall abc\,(Rabc \to Ra(\boldsymbol{s}b)(\boldsymbol{s}c)) \to Rmnk).$$

Siła wyrazu języka drugiego rzędu

Przykład: Dodawanie liczb naturalnych ($m + n = k$):

$$\forall R(\forall a(Ra0a) \to \forall abc\,(Rabc \to Ra(\boldsymbol{s}b)(\boldsymbol{s}c)) \to Rmnk).$$

Dodawanie jest najmniejszym punktem stałym operatora:

$$R \mapsto \{\langle a, 0, a\rangle \mid a \in \mathbb{N}\} \cup \{\langle a, \boldsymbol{s}b, \boldsymbol{s}c\rangle \mid \langle a, b, c\rangle \in R\}.$$

Siła wyrazu języka drugiego rzędu

Przykład: Dodawanie liczb naturalnych ($m + n = k$):

$$\forall R(\forall a(Ra0a) \to \forall abc\,(Rabc \to Ra(\boldsymbol{s}b)(\boldsymbol{s}c)) \to Rmnk).$$

Dodawanie jest najmniejszym punktem stałym operatora:

$$R \mapsto \{\langle a, 0, a\rangle \mid a \in \mathbb{N}\} \cup \{\langle a, \boldsymbol{s}b, \boldsymbol{s}c\rangle \mid \langle a, b, c\rangle \in R\}.$$

Uogólnienie: Najmniejszy punkt stały operatora Φ:

$$\mathrm{LFP}_\Phi(a) := \forall R((\Phi(R) \subseteq R) \to Ra)$$

$$\mathrm{LFP}_\Phi(a) := \forall R(\forall b(\Phi(R)b \to Rb) \to Ra)$$

Logika drugiego rzędu w stylu Henkina

Semantyka (nieformalna): Interpretujemy zmienne relacyjne jako *definiowalne* predykaty.
To ma sens klasycznie i intuicjonistycznie.

Reguły wnioskowania:

$$(\forall^2\mathsf{I})\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall R\,\varphi}\ (R \notin \mathsf{FV}(\Gamma)) \qquad (\forall^2\mathsf{E})\ \frac{\Gamma \vdash \forall R\,\varphi}{\Gamma \vdash \varphi[R := \lambda\!\!\lambda \vec{a}.\psi]}$$

$$(\exists^2\mathsf{I})\ \frac{\Gamma \vdash \varphi[R := \lambda\!\!\lambda \vec{a}.\psi]}{\Gamma \vdash \exists R\,\varphi} \qquad (\exists^2\mathsf{E})\ \frac{\Gamma \vdash \exists R\,\varphi \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}\ (R \notin \mathsf{FV}(\Gamma, \psi))$$

Brouwer-Heyting-Kołmogorow

- *A construction of* $\forall R\, \varphi(R)$ *is a method (function) transforming every predicate* $\mathbf{R}$ *into a proof of* $\varphi(\mathbf{R})$.
- *A construction of* $\exists R\, \varphi(R)$ *consists of a predicate* $\mathbf{R}$ *and a construction of* $\varphi(\mathbf{R})$.