Paweł Gładki

Logika matematyczna w informatyce

http://www.math.us.edu.pl/~pgladki/

Konsultacje: *Piątek, 8:00-9:30*

Jeżeli chcesz spotkać się z prowadzącym podczas konsultacji, postaraj się powiadomić go o tym przed lub po zajęciach, zadzwoń do jego pokoju, lub wyślij mu emaila.

Zasady zaliczania przedmiotu:

7 zadań domowych wartych w sumie 70 punktów i kolokwium warte 30 punktów.

Brak egzaminu

Kolokwium odbędzie się na przedostatnich zajęciach laboratoryjnych i będzie trwało 90 minut.

Zadania domowe:

Rozwiązania zadań domowych należy przesyłać przez email.

Dopuszczalne, a nawet pożądane jest organizowanie się w grupy do wspólnej nauki, ale rozwiązania muszą być indywidualne: prace, których autorstwo będzie budziło wątpliwości zostaną ocenione na 0 punktów.

Plan wykładu:

1. Rachunek zdań.
2. Rachunek predykatów. Semantyka formuł. Prawdziwość i spełnialność. Nierozstrzygalność.
3. Paradygmaty dowodzenia. Naturalna dedukcja. Rachunek sekwentów.
4. Logika intuicjonistyczna. Intuicjonistyczny rachunek zdań. Normalizacja dowodów.
5. Rachunek λ z typami prostymi. Izomorfizm Curry'ego-Howarda dla minimalnego intuicjonistycznego rachunku zdań.
6. Rachunek λ z typami zależnymi. Izomorfizm Curry'ego-Howarda-de Bruijna dla minimalnego intuicjonistycznego rachunku predykatów.
7. Obliczenia i wnioskowanie w systemie `Coq`.

Literatura:

1. Notatki z wykładów dostępne na stronie (w permanentnej konstrukcji...).
2. J. Tiuryn, J. Tyszkiewicz, P. Urzyczyn, *Logika dla informatyków*, `http://www.mimuw.edu.pl/~urzy/calosc.pdf`
3. M. Huth, M. Ryan, *Logic in Computer Science,* Cambridge, 2004.
4. P. Urzyczyn, *Rachunek lambda,* `http://www.mimuw.edu.pl/~urzy/Lambda/erlambda.pdf`
5. Y. Bertot, P. Casteran, *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*, Springer, 2004.
6. A. Chlipala, *Certified Programming with Dependent Types*, `http://adam.chlipala.net/cpdt/`.

Czym jest `Coq`?

Proof assistant/proof management system: narzędzie do interaktywnego dowodzenia twierdzeń:

- w języku specyfikacji możemy zapisać matematyczne definicje, twierdzenia, dowody, ale też wykonywalne programy funkcyjne z typami zależnymi;
- interaktywne środowisko dowodzenia twierdzeń pozwala na dowodzenie za pomocą taktyk;
- wspiera automatyzację dowodzenia.

Zastosowania `Coq`

Kto i do czego używa Proof assistants?
Informatycy używają ich do:

- weryfikacji fragmentów informatyki (weryfikacja dedukcyjna);
- formalizacji metateorii języków programowania;
- weryfikacji programów;
- generowania kodu (ekstrakcja z dowodu);
- współpracy z zewnętrznymi narzędziami do weryfikacji programów (*Why 3*, *Frama C*)

Projekty o znaczeniu nie tylko akademickim:

- *Popl mark challenge* czyli zmechanizowana metateoria dla mas: formalnie udowodnij metateorię dla twojego ulubionego języka programowania;
- *Cristal Project:* certyfikowany kompilator języka C CompCert;
- weryfikacja *Java Card*.

Matematycy używają ich to:

- konstruowania nowych teorii formalnych;
- weryfikowania dowodów.

Matematycy nie są głównymi użytkownikami tego typu oprogramowania:

- weryfikacja dowodów matematycznych przy użyciu Coq jest zbyt czasochłonna, jest to też proces głównie mechaniczny.

Preliminaria algebraiczne

Relacje i funkcje. Niech X i Y będą niepustymi zbiorami. Zbiór wszystkich par uporządkowanych (x, y), $x \in X$, $y \in Y$, nazywamy **produktem** zbiorów X i Y i oznaczamy:

$$X \times Y = \{(x, y) | x \in X, y \in Y\}.$$

Uwaga:

$$(x, y) = \{x, \{x, y\}\}.$$

Przez indukcję wprowadzamy pojęcie n-**ki uporządkowanej:**

$$(x_1, \ldots, x_n) = ((x_1, \ldots, x_{n-1}), x_n)$$

oraz **produktu** n **zbiorów**:

$$X_1 \times \ldots \times X_n = \{(x_1, \ldots, x_n) | x_1 \in X_1, \ldots, x_n \in X_n\}.$$

Dla uproszczenia notacji będziemy pisali

$$\prod_{i=1}^{n} X_i = X_1 \times \ldots \times X_n$$

oraz

$$X^n = \prod_{i=1}^{n} X.$$

Relacją ze zbioru X w zbiór Y nazywamy podzbiór

$$R \subset X \times Y.$$

Jeżeli $(x, y) \in R$, to zwyczajowo piszemy xRy. W przypadku, gdy $X = Y$, mówimy po prostu o relacji w zbiorze X. Dla danej relacji $R \subset X \times Y$ możemy zdefiniować **relację odwrotną** $R^{-1} \subset Y \times X$ w następujący sposób:

$$(y, x) \in R^{-1} \leftrightarrow (x, y) \in R.$$

Podobnie, jeśli $R \subset X \times Y$ oraz $S \subset Y \times Z$ są relacjami, to możemy zdefiniować **złożenie relacji** $S \circ R$**:**

$$(x, z) \in S \circ R \Leftrightarrow \exists y \in Y[(x, y) \in R \wedge (y, z) \in S].$$

Wśród relacji na zbiorze X wyróżniamy kilka szczególnych typów: mówimy, że relacja $R \subset X \times X$ jest

1. **zwrotna,** jeżeli dla $x \in X$ zachodzi xRx;
2. **symetryczna,** jeżeli dla $x, y \in X$ zachodzi $xRy \Rightarrow yRx$;
3. **antysymetryczna,** jeżeli dla $x, y \in X$ zachodzi $xRy \wedge yRx \Rightarrow x = y$;
4. **przechodnia,** jeżeli dla $x, y, z \in X$ zachodzi $xRy \wedge yRz \Rightarrow xRz$;
5. **spójna,** jeżeli dla $x, y \in X$ zachodzi $xRy \vee yRx$.

Funkcją ze zbioru X w zbiór Y nazywamy relację $f \subset X \times Y$ taką, że:

$$xfy \wedge xfy' \Rightarrow y = y'.$$

Funkcje $f \subset X \times Y$ zwykle oznaczamy jako $f : X \to Y$, zaś zamiast xfy piszemy na ogół $f(x) = y$. $f : X \to Y$ nazywamy

1. **injekcją** (albo **funkcją różnowartościową**), gdy jeśli $f(x_1) = f(x_2)$, to $x_1 = x_2$, dla $x_1, x_2 \in X$;
2. **surjekcją** (albo **funkcją "na"**), gdy dla każdego $y \in Y$ istnieje $x \in X$ taki, że $f(x) = y$;
3. **bijekcją,** gdy jest injekcją i bijekcją.

Do danej funkcji zawsze istnieje relacja odwrotna, ale nie zawsze musi ona być funkcją.

Stwierdzenie

Niech $f : X \to Y$ będzie funkcją. Wówczas f jest injekcją wtedy i tylko wtedy, gdy istnieje do niej funkcja odwrotna.

Jeżeli $s : X \to Y$ i $r : Y \to X$ są funkcjami takimi, że

$$s \circ r = \mathrm{id}_X,$$

to s nazywamy **sekcją,** a r **retrakcją.**

Wśród wszystkich relacji, jakimi będziemy się zajmować, wyszczególnimy dwa rodzaje: równoważności i porządki.
Równoważnością nazywamy relację w zbiorze X, która jest

1. zwrotna,
2. symetryczna oraz
3. przechodnia.

Jeżeli $x \in X$ oraz $\sim$ jest relacją równoważności w X, to zbiór wszystkich elementów, które są w relacji $\sim$ z x nazywamy **klasą abstrakcji** x (lub **klasą równoważności** x) i oznaczamy

$$[x]_\sim = \{y \in X : x \sim y\}.$$

Gdy będzie jasne z jaką relacją akurat pracujemy, to będziemy zwyczajnie pisać $[x]$ zamiast $[x]_\sim$. Zbiór wszystkich klas abstrakcji relacji $\sim$ oznaczamy przez

$$X/\sim = \{[x]_\sim : x \in X\}.$$

Wspomnijmy jeszcze o związku między relacjami równoważności a partycjami zbioru. **Partycją** (lub **podziałem**) zbioru X nazywamy rodzinę jego podzbiorów $\mathcal{P}$ taką, że:

1. $\forall P_1, P_2 \in \mathcal{P}(P_1 \neq P_2 \Rightarrow P_1 \cap P_2 = \emptyset)$ (a zatem P_1 i P_2 są parami rozłączne) oraz
2. $\bigcup\{P \in \mathcal{P}\} = X$ (a więc $\mathcal{P}$ jest pokryciem zbioru X).

Mamy następujące:

Stwierdzenie

Niech $\sim$ będzie relacją równoważności w zbiorze X. Wówczas $X/\sim$ jest partycją zbioru X. Na odwrót, każda partycja wyznacza relację równoważności.

Porządkiem w zbiorze X nazywamy relację, która jest:

1. zwrotna,
2. antysymetryczna oraz
3. przechodnia.

Algebrą nazywamy strukturę $\mathbf{A} = (A, \{F_i : i \in I\})$, gdzie A jest zbiorem zwanym **uniwersum algebry,** zaś $F_i : A^{\sharp F_i} \to A$ (symbol $\sharp F_i$ oznacza ilość argumentów funkcji F_i). W rozważanych przez nas algebrach I najczęściej będzie zbiorem skończonym. **Typem** (lub **sygnaturą**) algebry $\mathbf{A} = (A, \{F_i : i \in I\})$ nazywamy układ

$$\tau_{\mathbf{A}} = (\sharp F_i : i \in I).$$

Algebry **A** i **B** są **podobne,** gdy $\tau_{\mathbf{A}} = \tau_{\mathbf{B}}$. Jeżeli $\sigma : I \to \mathbb{N}$ oraz $\tau : J \to \mathbb{N}$ spełniają warunki

1. $I \supset J$,
2. $\sigma(a) = \tau(a)$ dla $a \in J$,

to σ nazywamy **wzbogaceniem** typu τ, a τ **reduktem** typu σ.

Ważnym przykładem algebr, który będziemy bliżej studiować są monoidy. **Monoidem** nazywamy algebrę $\mathbf{M} = (M, +, 0)$ o typie $\tau_M = (2, 0)$, dla której spełnione są następujące aksjomaty:

1. $x + 0 = 0 + x = x$ dla wszelkich $x \in M$ (tzn. 0 jest elementem neutralnym $+$) oraz
2. $x + (y + z) = (x + y) + z$ dla wszelkich $x, y, z \in M$ (tzn. $+$ jest łączne).

Jeżeli ponadto spełniony jest warunek

$$x + y = y + x$$

dla wszelkich $x, y \in M$, to $\mathbf{M}$ nazywamy **monoidem przemiennym** (lub **abelowym**).

Oznaczmy przez $\mathbb{K}(\tau)$ klasę wszystkich algebr typu τ. Niech $\mathbf{A} = (A, \{F_i : i \in I\})$ i $\mathbf{B} = (B, \{G_i : i \in I\})$ będą algebrami podobnymi. Mówimy, że $\mathbf{B}$ jest **podalgebrą** algebry $\mathbf{A}$ (oznaczamy $\mathbf{B} \subset \mathbf{A}$), gdy $B \subset A$ oraz dla każdego $i \in I$

$$G_i = F_i|_{B^{\sharp F_i}}$$

(symbol $|$ oznacza istotne zacieśnienie). Dalej, niech $X \subset B$. Mówimy, że X **generuje algebrę $\mathbf{B}$,** gdy $\mathbf{B}$ jest najmniejszą podalgebrą algebry $\mathbf{A}$ zawierającą uniwersum zawierające X.

Niech $\mathbf{A} = (A, \{F_i : i \in I\})$ i $\mathbf{B} = (B, \{G_i : i \in I\})$ będą algebrami podobnymi. Odwzorowanie $\phi : A \to B$ nazywamy **homomorfizmem** algebr $\mathbf{A}$ i $\mathbf{B}$, co oznaczamy przez $\phi : \mathbf{A} \to \mathbf{B}$, gdy dla każdego $i \in I$ i dla dowolnych $a_1, \ldots, a_n \in A$, gdzie $n = \sharp F_i$:

$$\phi(F_i(a_1, \ldots, a_n)) = G_i(\phi(a_1), \ldots, \phi(a_n)).$$

Monomorfizm jest to homomorfizm injektywny, **epimorfizm** to homomorfizm surektywny, a **izomorfizm** to homomorfizm bijektywny.

Odtąd tam, gdzie jest to konieczne, milcząco zakładamy podobieństwo algebr $\mathbf{A}$ i $\mathbf{B}$. Niech $\mathbb{K}$ oznacza klasę algebr podobnych. Spośród licznych konstrukcji na algebrach wyróżnimy trzy: algebry wolne, ilorazowe i produkty algebr. Mówimy, że $\mathbf{A}$ jest **algebrą wolną w $\mathbb{K}$ ze zbiorem wolnych generatorów X,** gdy X generuje $\mathbf{A}$ oraz dla każdej algebry $\mathbf{B} \in \mathbb{K}$ i dowolnego odwzorowania $\phi : X \to B$ istnieje dokładnie jedno przedłużenie ϕ do homomorfizmu algebr $\tilde{\phi} : \mathbf{A} \to \mathbf{B}$. Algebrę $\mathbf{B}$ nazywamy po prostu **wolną**, gdy jest algebrą wolną w klasie wszystkich algebr podobnych do $\mathbf{A}$.

Niech $\mathbf{A} = (A, \{F_i : i \in I\})$. **Kongruencją** algebry $\mathbf{A}$ nazywamy relację $R \subset A \times A$ taką, że

1. R jest relacją równoważności,
2. dla każdego $i \in I$ i dla dowolnych $a_1, \ldots, a_n \in A$, gdzie $n = \sharp F_i$:

$$\text{jeżeli } a_1 R b_1, \ldots, a_n R b_n, \text{ to } F_i(a_1, \ldots, a_n) R G(b_1, \ldots, b_n).$$

Niech $\mathbf{A} = (A, \{F_i : i \in I\})$ i niech R będzie kongruencją algebry $\mathbf{A}$. **Algebrą ilorazową** algebry $\mathbf{A}$ nazywamy algebrę

$$\mathbf{A}/R = (A/R, \{F_i^R : i \in I\}),$$

gdzie

$$F_i^R([a_1], \ldots, [a_n]) = [F_i(a_1, \ldots, a_n)],$$

dla każdego $i \in I$ i dla dowolnych $a_1, \ldots, a_n \in A$, gdzie $n = \sharp F_i$. Epimorfizm $\kappa : \mathbf{A} \to \mathbf{A}/R$ dany wzorem

$$\kappa(a) = [a]$$

zwiemy **epimorfizmem kanonicznym.**

Na koniec niech $\{\mathbf{A}_t : t \in T\}$, przy czym $\mathbf{A}_t = (A_t, \{F_i^t : i \in I\})$, będzie rodziną algebr podobnych. **Produktem** tej rodziny nazywamy algebrę

$$\prod_{t\in T} \mathbf{A}_t = (\prod_{t\in T} A_t, \{G_i : i \in I\})$$

gdzie

$$G_i((a_t^1)_t, \ldots, (a_t^n)_t) = (F_i^t(a_t^1, \ldots, a_t^n))_t$$

dla dowolnych $i \in I$, $a_t^1, \ldots, a_t^n$ i $n = \sharp F_i$.