Logika i teoria typów

Wykład 1

5 października 2022

Po co teoria typów?

- Motywacja „filozoficzna": Bo to alternatywny (w stosunku do teorii mnogości) sposób patrzenia na matematykę.
- Motywacja „techniczna": Bo to np. język narzędzi wspomagających dowodzenie, jak np. Coq.

O czym będzie ten wykład:

O różnych systemach logicznych,
w których typy odgrywają istotną rolę.

Punkt widzenia: Izomorfizm Curry'ego-Howarda

- Formuła = typ = specyfikacja;
- Dowód = program = implementacja;
- Normalizacja dowodu = obliczenie.
- Konstrukcja dowodu = obliczenie.

Klasyczny slogan: *Proofs into programs!*

Nowy slogan: *Programs into proofs!*

O czym będzie ten wykład:

- Powtórzenie z rachunku lambda (turbo).
- Logika intuicjonistyczna.
- Logika jako gra dialogowa.
- Podstawy logiki liniowej.
- Logika klasyczna, kontynuacje i wyjątki.
- Polimorfizm.
- Typy zależne.
- Rachunek konstrukcji i jego uogólnienia.
- Co to jest Coq?
- Typy indukcyjne i rekurencyjne.
- Homotopijna teoria typów.

Wykład: https://us02web.zoom.us/j/84593508534

Konsultacje PU: czwartki, 19–20:30, tamże.

Moodle:
https://moodle.mimuw.edu.pl/course/view.php?id=1488

Powtórzenie z rachunku lambda

Zbiory i funkcje

Sposób użycia:
$a \in A$ (należenie) $F(a)$ (aplikacja)

Tworzenie:
$\{x \mid W(x)\}$ (wycinanie) $\lambda x\, W(x)$ (abstrakcja)

Ewaluacja:
$a \in \{x \mid W(x)\} \quad \Leftrightarrow \quad W(a) \qquad (\lambda x\, W(x))(a) = W(a)$

Ekstensjonalność (?)

Dla zbiorów (niewątpliwa):

$A = B$ wtedy i tylko wtedy, gdy $\forall x\,(x \in A \Leftrightarrow x \in B)$

$$A = \{x \mid x \in A\}$$

Dla funkcji (wątpliwa):

$F = G$ wtedy i tylko wtedy, gdy $\forall x\,(F(x) = G(x))$

$$F = \lambda x\, Fx \quad ([1])$$

[1] Gdy F nie zawiera x.

Beztypowy rachunek lambda

Lambda-wyrażenia:

- Zmienne x, y, z, ...
- Aplikacje (MN);
- Abstrakcje $(\lambda x\, M)$.

Konwencje:

- Opuszczamy zewnętrzne nawiasy;
- Aplikacja wiąże w lewo: MNP oznacza $(MN)P$
- Skrót z kropką: $\lambda x_1 \ldots x_n.M$ oznacza $\lambda x_1(\ldots(\lambda x_n M)\cdots)$.

Przykłady

$$
\begin{aligned}
\mathsf{I} &= \lambda x.x \\
\mathsf{K} &= \lambda xy.x \\
\mathsf{S} &= \lambda xyz.xz(yz) \\
2 &= \lambda fx.\, f(fx) \\
\omega &= \lambda x.xx \\
\Omega &= \omega\omega \\
\mathsf{Y} &= \lambda f((\lambda x.f(xx))(\lambda x.f(xx))
\end{aligned}
$$

Zmienne *wolne* (globalne)

$\mathrm{FV}(x) = \{x\}$;
$\mathrm{FV}(MN) = \mathrm{FV}(M) \cup \mathrm{FV}(N)$;
$\mathrm{FV}(\lambda x\, M) = \mathrm{FV}(M) - \{x\}$.

Na przykład:

$\mathrm{FV}(\lambda x\, x) = \varnothing$;
$\mathrm{FV}(\lambda x.\, xy) = \{y\}$;
$\mathrm{FV}((\lambda x.\, xy)(\lambda y.\, xy)) = \{x, y\}$.

Alfa-konwersja

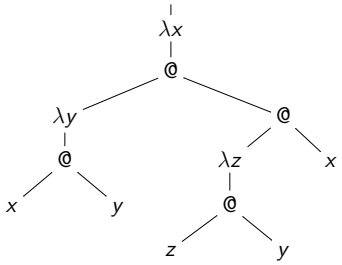
Wyrażenia $\lambda x.\, xy$ i $\lambda z.\, zy$ oznaczają tę samą operację („zaaplikuj dany argument do y").

Należy je uważać za identyczne.

Alfa-konwersja: Wyrażenia różniące się tylko wyborem zmiennych związanych utożamiamy.

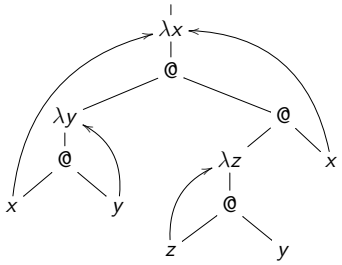
Lambda-termy to klasy abstrakcji tego utożsamienia.

Lambda-wyrażenie jako drzewo



$\lambda x.(\lambda y.\, xy)((\lambda z.\, zy)x)$

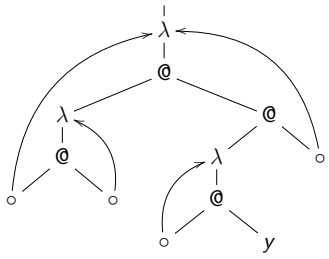
Lambda-wyrażenie jako graf



$\lambda x.(\lambda y.\, xy)((\lambda z.\, zy)x)$

Lambda-term jako graf

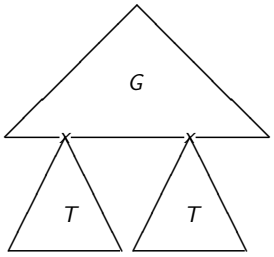
Zmienne związane sa niepotrzebne.



$\lambda x.(\lambda y.\, xy)((\lambda z.\, zy)x)$

Podstawienie $G[x := T]$

Podstawienie termu T do termu G w miejsce wolnych wystąpień zmiennej x.



Podstawienie

- $x[x := N] = N$;
- $y[x := N] = y$, gdy y jest zmienną różną od x;
- $(PQ)[x := N] = P[x := N]Q[x := N]$;
- $(\lambda y\, P)[x := N] = \lambda y.P[x := N]$,
  gdy $y \neq x$ oraz $y \notin \mathrm{FV}(N)$.

Wykonanie podstawienia na konkretnej reprezentacji termu może wymagać wymiany zmiennych:

$(\lambda y\, P)[x := N] = \lambda z\, P[y := z][x := N]$, gdzie z jest „nowe".

Beta-redukcja

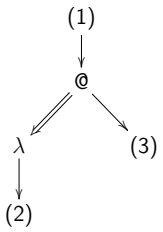
Najmniejsza relacja $\to_\beta$, spełniająca warunki:

- $(\lambda xP)Q \to_\beta P[x := Q]$;
- jeśli $M \to_\beta M'$, to:
  $MN \to_\beta M'N$, $NM \to_\beta NM'$ oraz $\lambda xM \to_\beta \lambda xM'$.

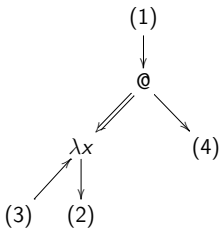
Term postaci $(\lambda xP)Q$ to *β-redeks*.

Relacja $\to_\beta$ to zredukowanie jednego dowolnego redeksu.

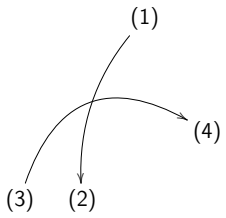
Dygresja: beta-redeks to krawędź w grafie



Przed redukcją i po redukcji



$(\lambda xP)Q$



$P[x := Q]$;

Wołanie przez nazwę

$$(\lambda xP)Q \to_\beta P[x := Q]$$

Ewaluacja procedury o parametrze formalnym x i treści P, gdy parametrem aktualnym jest Q:

Należy wstawić parametr aktualny do treści procedury, wymieniając, jeśli trzeba, lokalne identyfikatory na nowe.

Relacje pochodne:

Dowolna liczba kroków: $\twoheadrightarrow_\beta$ lub $\to_\beta^*$;

Niezerowa liczba kroków: $\to_\beta^+$;

Co najwyżej jeden krok: $\to_\beta^=$;

Równoważność (beta-konwersja): $=_\beta$.

Przykład: SKK $=_\beta$ I

$$\begin{aligned}
\mathsf{SKK} &= (\lambda xyz.xz(yz))(\lambda xy.x)(\lambda xy.x)\\
&\to_\beta (\lambda yz.(\lambda xy.x)z(yz))(\lambda xy.x)\\
&\to_\beta \lambda z.(\lambda xy.x)z((\lambda xy.x)z)\\
&\to_\beta \lambda z.(\lambda y.z)((\lambda xy.x)z)\\
&\to_\beta \lambda z.(\lambda y.z)(\lambda y.z)\\
&\to_\beta \lambda z.z = \mathsf{I}
\end{aligned}$$

Postaci normalne

Postać normalna to term bez redeksów.

Nie da się go redukować.

Termy w postaci β-normalnej są takie:

- Abstrakcje: $\lambda x\, M$, (gdzie M normalny);
- Eliminatory: $x\, N_1 \ldots N_k$, (gdzie $N_1, \ldots, N_k$ normalne).

Inaczej: termy postaci $\lambda x_1 \ldots x_n.yN_1 \ldots N_k$.

Normalizacja

Term M *ma postać normalną* (*jest normalizowalny*), gdy redukuje się do pewnej postaci normalnej.

Nazywamy ją *postacią normalną* termu M.

Term M jest *silnie normalizowalny* ($M \in \mathrm{SN}$), gdy nie istnieje nieskończony ciąg

$$M = M_0 \to_\beta M_1 \to_\beta M_2 \to_\beta \cdots$$

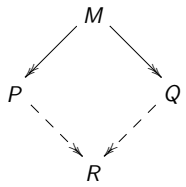
Inaczej: każdy ciąg redukcji prowadzi do postaci normalnej.

Przykłady

- Term $\mathsf{S} = \lambda xyz.xz(yz)$ jest w postaci normalnej.
- Term SKK jest silnie normalizowalny i ma postać normalną I.
- Term $\mathbf{\Omega} = (\lambda x.\, xx)(\lambda x.\, xx)$ nie ma postaci normalnej.
- Term $(\lambda x.\, y)\mathbf{\Omega}$ ma postać normalną y, ale nie jest silnie normalizowalny.

Twierdzenie Churcha-Rossera (CR)

Jeśli $M \twoheadrightarrow P$ i $M \twoheadrightarrow Q$, to istnieje takie R, że $P \twoheadrightarrow R$ i $Q \twoheadrightarrow R$.



Wniosek: *Jeśli $M =_\beta N$, to $M \twoheadrightarrow_\beta Q_\beta \twoheadleftarrow N$, dla pewnego Q.*

Wniosek: *Każdy term ma co najwyżej jedną postać normalną.*

Eta-reduction

The least relation $\to_\eta$, satisfying the conditions:

- $\lambda x.Mx \to_\eta M$, when $x \notin \mathrm{FV}(M)$;
- jeśli $M \to_\eta M'$, to $MN \to_\eta M'N$, $NM \to_\eta NM'$ oraz $\lambda xM \to_\eta \lambda xM'$.

Relacja $\to_{\beta\eta}$ to suma $\to_\beta$ i $\to_\eta$.

Notacja $\twoheadrightarrow_{\beta\eta}$, $=_{\beta\eta}$ itd. stosuje się odpowiednio.

Eta-redeks i beta-redeks

$\lambda x.\, Px$ $\qquad$ $(\lambda x\, P)Q$

Siła wyrazu

Curry's fixed point combinator $\mathbf{Y}$

$$\mathsf{Y} = \lambda f((\lambda x.f(xx))(\lambda x.f(xx))$$

Fact: $\mathsf{Y}F =_\beta F(\mathsf{Y}F)$, for every F.

Proof: $\mathsf{Y}F \to_\beta (\lambda x.F(xx))(\lambda x.F(xx)) \to_\beta F((\lambda x.F(xx))(\lambda x.F(xx))) \;{}_\beta\!\leftarrow F(\mathsf{Y}F)$

$\mathsf{Y}F =_\beta F(\mathsf{Y}F)$

Example: Find an M such that $Mxy =_\beta MxxyM$.

Solution: No problem, $M = \mathsf{Y}(\lambda m\, \lambda xy.\, mxxym)$.

Conditional

$$\text{true} = \lambda xy.x \qquad \text{false} = \lambda xy.y$$

$$\text{if } P \text{ then } Q \text{ else } R = PQR.$$

It works:

$$\text{if true then } Q \text{ else } R \twoheadrightarrow_\beta Q$$

$$\text{if false then } Q \text{ else } R \twoheadrightarrow_\beta R.$$

Ćwiczenie

Zdefiniować klasyczne spójniki zdaniowe: $\vee$, $\wedge$, $\to$, $\neg$.

Ordered pair

Pair = Boolean selector:

$$\begin{aligned} \langle M, N\rangle &= \lambda x.xMN; \\ \pi_i &= \lambda x_1 x_2.x_i \qquad (i = 1, 2); \\ \Pi_i &= \lambda p.\, p\pi_i \qquad (i = 1, 2). \end{aligned}$$

It works:

$$\Pi_1\langle M, N\rangle \to_\beta \langle M, N\rangle\pi_1 \twoheadrightarrow_\beta M.$$

Church's numerals

$$c_n = \mathrm{n} = \lambda fx.f^n(x),$$

$$\begin{aligned} 0 &= \lambda fx.x; \\ 1 &= \lambda fx.fx; \\ 2 &= \lambda fx.f(fx); \\ 3 &= \lambda fx.f(f(fx)), \text{ etc.} \end{aligned}$$

Definable functions

A partial function $f : \mathbb{N}^k \multimap \mathbb{N}$ is *λ-definable* if there is a term F such that for all $n_1, \ldots, n_k \in \mathbb{N}$:

- If $f(n_1, \ldots, n_k) = m$, then $F\mathrm{n}_1 \ldots \mathrm{n}_k =_\beta \mathrm{m}$;
- If $f(n_1, \ldots, n_k)$ is undefined then $F\mathrm{n}_1 \ldots \mathrm{n}_k$ does not normalize.

Nierozstrzygalność

Twierdzenie: *Funkcja (częściowa) jest definiowalna w rachunku lambda wtedy i tylko wtedy, gdy jest (częściowo) rekurencyjna.*

Wnioski: The following are undecidable problems:

- Given M and N, does $M \twoheadrightarrow_\beta N$ hold?
- Given M and N, does $M =_\beta N$ hold?
- Given M, does M normalize?
- Given M, does M strongly normalize?

Typy proste

(Turbo-powtórzenie)

Simple types (Curry style)

Types:

- Zmienne i/lub stałe typowe, np. jedna stała 0.
- If σ and τ are types then $(\sigma \to \tau)$ is a type.

Konwencja:

- Zamiast $(\tau \to (\sigma \to \rho))$ piszemy $\tau \to \sigma \to \rho$.

Każdy typ ma postać $\tau_1 \to \cdots \to \tau_n \to \mathit{atom}$.

Environment (basis, context)

Environment: a set Γ of *declarations* of the form $(x : \tau)$
such that if $(x, \tau), (x, \sigma) \in \Gamma$ then $\tau = \sigma$.
Write $\Gamma(x) = \tau$.

Notation $\Gamma(x : \tau) = \Gamma \cup \{(x : \tau)\}$, when $x \notin \mathrm{Dom}(\Gamma)$.
Otherwise $\Gamma(x : \tau) = (\Gamma - \{x : \Gamma(x)\}) \cup \{(x : \tau)\}$.

Simple type assignment

$$\Gamma\,(x : \sigma) \vdash x : \sigma \ \text{(Var)}$$

$$\frac{\Gamma(x : \sigma) \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau} \ \text{(Abs)}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \ \text{(App)}$$

Note: This is a syntax-oriented system.

Przykłady

- $x : p \to q \to r,\ y : p \to q,\ z : p \ \vdash \ xz(yz) : r$;
- $\vdash \ \lambda xyz.\, xz(yz) : (p \to q \to r) \to (p \to q) \to \ p \to r$;
- $\lambda xy.\, x : p \to q \to p$;
- $\lambda xy.\, x : \tau \to \sigma \to \tau$;
- $2 : (\tau \to \tau) \to \tau \to \tau$;
- $\not\vdash \ \lambda x.\, xx : \tau$, dla każdego τ.

Type reconstruction

Theorem: *Typability* $(\,? \vdash M : ?\,)$ *is decidable in* PTIME.

Proof: Reduction to unification.

Fact: *The same holds for type-checking* $(\Gamma \vdash M : \tau?)$

Proof: Similar.

Subject Reduction

Theorem:
If $\Gamma \vdash M : \tau$ *and* $M \to_{\beta\eta} N$ *then* $\Gamma \vdash N : \tau$.

Silna normalizacja

Twierdzenie:
Jeśli $\Gamma \vdash M : \tau$, *to term* M *jest silnie normalizowalny.*

Church vs. Curry

Curry style (type-assignment systems):

- Ordinary untyped lambda-terms.
- Types are derivable properties of terms.
- System of type assignment rules.
- A term may have many types or none.
- Typability not obvious.

Church style (typed systems):

- New syntax, built-in types.
- Every term has exactly one type.
- No "untypable" terms.

Non-orthodox Church

Type-assignment with type annotations on bound variables.

$$\Gamma\,(x : \sigma) \vdash x : \sigma \ \text{(Var)}$$

$$\frac{\Gamma(x : \sigma) \vdash M : \tau}{\Gamma \vdash \lambda x{:}\sigma\, M : \sigma \to \tau} \ \text{(Abs)}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \ \text{(App)}$$

Fact: If $\Gamma \vdash M : \tau$ and $\Gamma \vdash M : \sigma$ then $\tau = \sigma$.

Relating systems

Erasing types from (non-orthodox) Church terms:

- $|x| = x$;
- $|MN| = |M||N|$;
- $|\lambda x{:}\sigma.\, M| = \lambda x\, |M|$.

Properties of type erasure

(Church is blue, Curry is black.)

1. If $\Gamma \vdash \mathrm{M} : \tau$ then $\Gamma \vdash |\mathrm{M}| : \tau$.
2. If $\Gamma \vdash M : \tau$ then there exists M such that $|\mathrm{M}| = M$ oraz $\Gamma \vdash \mathrm{M} : \tau$.
3. If $\mathrm{M} \to_\beta \mathrm{N}$ then $|\mathrm{M}| \to_\beta |\mathrm{N}|$.
4. If $|\mathrm{M}| \to_\beta N$ then there exists N such that $|\mathrm{N}| = N$ and $\mathrm{M} \to_\beta \mathrm{N}$.

Proof: easy induction.

Example: If $M = |\mathrm{M}|$ then $M \in \mathrm{SN}$ iff $\mathrm{M} \in \mathrm{SN}$.

Informal type annotations

Typable Curry style terms can be informally annotated by types, e.g. $((\lambda x^\sigma.N^\tau)^{\sigma\to\tau}P^\sigma)^\tau$. Such annotations represent type derivations and can be identified with Church-style terms.

In many cases it does not matter if we consider Curry style or Church style, orthodox or not. We always choose what is the most convenient formulation.

N.B. Coq is non-orthodox Church

Principal pair

A pair (Γ, τ) is a *principal pair* for M iff the following are equivalent for all Γ' and τ':

- $\Gamma' \vdash M : \tau'$;
- $S(\Gamma) \subseteq \Gamma'$ and $S(\tau) = \tau'$, for some substitution S.

Then τ is the *principal type* of M.

Corollary: If a term M is typable, then there exists a principal pair for M. This principal pair is unique up to renaming of type variables.

Examples

- Principal type of **S** is $(p \to q \to r) \to (p \to q) \to p \to r$. Another non-principal type of **S**:
  $$(p \to q \to p) \to (p \to q) \to p \to p$$
- Type $\omega = (p \to p) \to p \to p$ is the principal type of Church numerals n, for $n \geq 2$. For **0** and **1** the principal types are respectively $p \to q \to q$ and $(p \to q) \to p \to q$. Every Church numeral can also be assigned the type $\omega_{p\to q} = ((p \to q) \to p \to q) \to (p \to q) \to p \to q$.

Logika kombinatoryczna (Rachunek kombinatorów)

Terms:

- Variables;
- Constants K and S;
- Applications (FG).

The "weak" reduction

- $KFG \to_w F$;
- $SFGH \to_w FH(GH)$;
- If $F \to_w G$, then $FH \to_w GH$ and $HF \to_w HG$.

Notation $\twoheadrightarrow_w$, $=_w$ etc. used as usual.

Examples of combinators

- $\mathsf{I} = \mathsf{SKK}$
  $\mathsf{I}F \to_w \mathsf{K}F(\mathsf{K}F) \to_w F$
- $\Omega = \mathsf{SII}(\mathsf{SII})$
  $\Omega \to_w \mathsf{I}(\mathsf{SII})(\mathsf{I}(\mathsf{SII})) \twoheadrightarrow_w \Omega$
- $\mathsf{W} = \mathsf{SS}(\mathsf{KI})$
  $\mathsf{W}FG \to_w \mathsf{S}F(\mathsf{KI}F)G \to_w \mathsf{S}F\mathsf{I}G \to_w FG(\mathsf{I}G) \twoheadrightarrow_w FGG$

Examples of combinators

- $\mathsf{B} = \mathsf{S}(\mathsf{KS})\mathsf{K}$

  $\mathsf{B}FGH \to_w \mathsf{KS}F(\mathsf{K}F)GH \to_w \mathsf{S}(\mathsf{K}F)GH \to_w$
  $\to_w \mathsf{K}FH(GH) \to_w F(GH)$

- $\mathsf{C} = \mathsf{S}(\mathsf{BBS})(\mathsf{KK})$

  $\mathsf{C}FGH \twoheadrightarrow_w FHG$

- $\mathsf{B}' = \mathsf{CB}$

  $\mathsf{B}'FGH \twoheadrightarrow_w G(FH)$

Remark: Each of those can be added as a new constant.

Examples of combinators

- $0 = \mathsf{KI}$ $\quad 0FG \twoheadrightarrow_w G$
- $1 = \mathsf{SB}(\mathsf{KI})$ $\quad 1FG \twoheadrightarrow_w FG$
- $2 = \mathsf{SB}(\mathsf{SB}(\mathsf{KI}))$ $\quad 2FG \twoheadrightarrow_w F(FG)$

From lambda to CL: combinatory abstraction

- $\lambda^* x.F = \mathsf{K}F$, when $x \notin \mathrm{FV}(F)$;
- $\lambda^* x.x = \mathsf{I}$;
- $\lambda^* x.FG = \mathsf{S}(\lambda^* x.F)(\lambda^* x.G)$, otherwise.

Fakt: $(\lambda^* x.F)G \twoheadrightarrow_w F[x := G]$.

Fixed point combinators

$$\mathsf{Y} = \mathsf{WS}(\mathsf{BWB})$$

$$\Theta = \mathsf{WI}(\mathsf{B}(\mathsf{SI})(\mathsf{WI}))$$

NCL - "Naive Combinatory Logic"

Terms: If F, G are terms then $\mathsf{P}FG$ is a term.
Write $F \Rightarrow G$ for $\mathsf{P}FG$.

Formulas:
– Every term F is a formula;
– Equations $F = G$ are formulas.

NCL - Axioms and rules

Axioms:

$F = F$ $\qquad \mathsf{K}FG = F$ $\qquad \mathsf{S}FGH = FH(GH)$

$F \Rightarrow F$ $\qquad (F \Rightarrow (F \Rightarrow G)) \Rightarrow (F \Rightarrow G)$

Rules:

$$\frac{M = N}{MQ = NQ} \qquad \frac{M = N}{QM = QN}$$

$$\frac{M = N}{N = M} \qquad \frac{M = N,\ N = Q}{M = Q}$$

$$\frac{M, \quad M = N}{N} \qquad \frac{M, \quad M \Rightarrow N}{N}$$

Curry's Paradox

Take any term F and define $N = \mathsf{Y}(\lambda^* x.\ x \Rightarrow F)$.

Then $N = N \Rightarrow F$ in NCL.

Using axiom $N \Rightarrow N$ it follows that $N \Rightarrow (N \Rightarrow F)$.

Thus $N \Rightarrow F$, using $(N \Rightarrow (N \Rightarrow F)) \Rightarrow (N \Rightarrow F)$.

This proves N, because $N = N \Rightarrow F$.

Eventually, F follows by modus ponens.

Moral: System NCL is logically inconsistent.