**Uniwersytet Warszawski**
Wydział Matematyki, Informatyki i Mechaniki

**Agnieszka Kozubek**
Nr albumu: 197879

# Problem sprawdzania typu dla logiki pierwszego rzędu

**Praca magisterska**
**na kierunku INFORMATYKA**

Praca wykonana pod kierunkiem
**prof. dra hab. Pawła Urzyczyna**
Instytut Informatyki

Maj 2006

## Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

## Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

## Streszczenie

W literaturze rozważa się różne systemy typów dla rachunku lambda, będące zazwyczaj rozszerzeniami systemu typów prostych. W tej pracy rozważamy rozszerzenie prowadzące do logiki pierwszego rzędu. Badamy problem sprawdzania typu dla systemu zawierającego tylko implikację i kwantyfikator ogólny. Dowodzimy rozstrzygalności tego problemu dla sygnatury bez symboli funkcyjnych o arności większej od zera oraz szacujemy złożoność podanego algorytmu.



## Słowa kluczowe

sprawdzanie typu, logika pierwszego rzędu, polimorfizm, rachunek lambda

## Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

## Klasyfikacja tematyczna

F. Theory of Computation
F.4 Mathematical Logic and Formal Languages
F.4.1 Mathematical Logic
Lambda calculus and related systems


## Tytuł pracy w języku angielskim

Type checking for first order logic

*Mojej Mamie,*
*której nie było dane zobaczyć tej pracy*
*i mojemu Tacie,*
*który zawsze jest dla mnie oparciem*

# Spis treści

# Wprowadzenie

W literaturze rozważa się różne systemy typów dla rachunku lambda. Podstawowym systemem jest tu rachunek lambda z typami prostymi. Inne systemy typów zazwyczaj są rozszerzeniami systemu typów prostych. Rachunek typów prostych pozostaje w bliskim związku z intuicjonistycznym rachunkiem zdań. Naturalne jest zatem badanie takich systemów typów, które prowadzą do różnych rozszerzeń rachunku zdań. Na przykład bardzo ważny system typów, system $F$, to rachunek zdań drugiego rzędu. W tej pracy zajmiemy się innym rozszerzeniem rachunku zdań, logiką pierwszego rzędu. Będziemy badać implikacyjny fragment logiki pierwszego rzędu.

System podobny do badanego w tej pracy jest rozważany przez Sørensena i Urzyczyna w [5]. Wprowadzony tam system $\lambda P_1$ jest sformułowany w *stylu Churcha*. Lambda termy nie są termami czystego rachunku lambda, lecz przechowują pewną informację o typie termu. Ponadto Sørensen i Urzyczyn rozważają system dla pełnostrukturalnej logiki pierwszego rzędu, ze wszystkimi spójnikami logicznymi i stałą $\bot$. W tej pracy zajmiemy się systemem ograniczonym do fragmentu zawierającego tylko implikację i kwantyfikator ogólny. Badany system jest systemem w *stylu Curry'ego*. Lambda termy są termami czystego rachunku lambda, informacja o typie termu nie jest przechowywana w termie, lecz jest od niego oddzielona. Warto zauważyć, że system w stylu Curry'ego dla logiki pierwszego można zdefiniować inaczej, tak jak robi to Schubert w [4]. W tym systemie zachowujemy w lambda termie informację o operacjach pierwszego rzędu: wprowadzeniu lub usunięciu kwantyfikatora oraz użytym podstawieniu, usuwamy tylko informację o typie zmiennej związanej. Jest to rozwiązanie pośrednie między systemem rozważanym w tej pracy i systemem w stylu Churcha.

W każdym systemie typów dla rachunku lambda można postawić następujące pytania:

- Dany term rachunku lambda $M$, czy istnieją środowisko $\Gamma$ i typ $\tau$ takie, że $\Gamma \vdash M : \tau$ jest wyprowadzalne?
- Dany term rachunku lambda $M$, typ $\tau$ i środowisko $\Gamma$, czy $\Gamma \vdash M : \tau$ jest wyprowadzalne?

Pierwszy problem nazywa się problemem *typowalności*, a drugi problemem *sprawdzania typu*. Dla różnych systemów typów problemy te mają różną trudność. W systemie typów prostych ([3], [5]) oba problemy są rozstrzygalne. Wells w [7] pokazał, że w systemie $F$ typowalność i sprawdzanie typu są nierozstrzygalne. Dla logiki pierwszego rzędu problem typowalności jest rozstrzygalny. Schubert w [4] udowodnił, że dla rozważanej przez niego wersji pokrewny problem *wyprowadzania typu* jest rozstrzygalny. W tej pracy badamy problem sprawdzania typu dla logiki pierwszego rzędu.

Głównym rezultatem pracy jest stwierdzenie, że problem sprawdzania typu dla systemu bez wieloargumentowych symboli funkcyjnych jest rozstrzygalny. Dowodzimy, że jeżeli istnieje jakieś rozwiązanie, to można również znaleźć rozwiązanie w pewnej skończonej klasie wyprowadzeń. Zatem aby stwierdzić, czy dana asercja jest wyprowadzalna, wystarczy sprawdzić

skończoną liczbę możliwości. Ponadto podajemy ograniczenie górne na liczbę przypadków, które trzeba sprawdzić i szacujemy złożoność podanego algorytmu. Pozostaje do zbadania problem sprawdzania typu dla systemu z sygnaturą dopuszczającą wieloargumentowe symbole funkcyjne.

W rozdziale pierwszym przedstawiam definicje podstawowych pojęć, wprowadzam oznaczenia oraz formułuję w sposób formalny problem, którym będziemy się zajmować. Rozdział drugi zawiera dowód, że problem sprawdzania typu dla systemu z sygnaturą bez wieloargumentowych symboli funkcyjnych jest rozstrzygalny oraz oszacowanie złożoności podanego algorytmu.

# Rozdział 1

# Podstawowe definicje i notacja

## 1.1. Notacja

- Przez $\mathbb{N}$ będę oznaczać zbiór liczb naturalnych.
- Niech $\Sigma$ będzie dowolnym zbiorem. Przez $\Sigma^*$ będę oznaczać zbiór słów skończonych nad alfabetem $\Sigma$. Jeżeli $a$ i $b$ są dwoma słowami nad tym samym alfabetem, to $a \cdot b$ oznacza konkatenację słów $a$ i $b$. Zamiast $a \cdot b$ będę często pisać $ab$.
- Niech $L$, $P$ będą zbiorami słów nad tym samym alfabetem $\Sigma$. Wtedy $L \cdot P = \{v \cdot w | v \in L \wedge w \in P\}$. Jeżeli $L = \{v\}$, to zamiast $\{v\} \cdot P$ będę pisać $v \cdot P$.
- Przez $f : A \rightharpoonup B$ będę oznaczać funkcję częściową $f$ ze zbioru $A$ do zbioru $B$.

## 1.2. Grafy

Będę rozważać skończone grafy nieskierowane o etykietowanych krawędziach. Jak zwykle, graf to para $(V, E)$, gdzie $V$ jest zbiorem wierzchołków, $E$ jest zbiorem krawędzi. Zbiór wierzchołków grafu $G$ będę oznaczać $V(G)$, zbiór krawędzi będę oznaczać $E(G)$. Przez $(x \overset{l}{\longleftrightarrow} y)$ będę oznaczać krawędź o etykiecie $l$ łączącą wierzchołki $x$ i $y$.

*Sumą rozłączną* indeksowanej rodziny grafów $(G_i)_{i \in I}$ będę nazywać graf $G = (V, E)$ taki, że:

- $V = \bigcup_{i \in I} \Big( \{i\} \times V_i \Big)$,
- jeśli $(a \overset{l}{\longleftrightarrow} b) \in E_i$, to $((i, a) \overset{l}{\longleftrightarrow} (i, b)) \in E$.

## 1.3. Lambda termy

Lambda termy definiujemy jak zazwyczaj, przyjmując zwykłe konwencje dotyczące ich zapisu za Barendregtem [1]. Zbiór zmiennych termowych będę oznaczać przez $V$, zbiór lambda termów będę oznaczać przez $\Lambda$. Lambda termy będę oznaczać dużymi literami $M$, $N$, $L$, zmienne termowe będę oznaczać literami $x$, $y$, $z$, $u$, $v$, itp. Symbol $FV(M)$ oznacza zbiór zmiennych wolnych termu $M$, $BV(M)$ oznacza zbiór zmiennych związanych termu $M$. Zakładam, że każda zmienna związana termu $M$ jest w nim związana co najwyżej jeden raz (tak dobieram reprezentanta, aby nie było problemu z alfa-konwersją). Zbiór wszystkich zmiennych termu $M$ będę oznaczać przez $V(M)$.

Zbiór wszystkich podtermów lambda termu $M$ będę oznaczać przez $SubT(M)$:

- $M \in SubT(M)$,
- jeśli $N_1N_2 \in SubT(M)$, to $N_1 \in SubT(M)$ i $N_2 \in SubT(M)$,
- jeśli $\lambda x.N \in SubT(M)$, to $x \in SubT(M)$ i $N \in SubT(M)$.

Zauważmy, że nie utożsamiamy alfa-równoważnych podtermów. Na przykład mamy $SubT((\lambda x.x)(\lambda y.y)) = \{x, y, \lambda x.x, \lambda y.y, (\lambda x.x)(\lambda y.y)\}$, ale jednocześnie $SubT((xy)(xy)) = \{x, y, xy, (xy)(xy)\}$. Podkreślmy, że w drugim przykładzie $x$, $y$ i $xy$ występują w zbiorze $SubT$ tylko jeden raz.

## 1.4. Typy

Typy będę oznaczać literami $\tau$, $\sigma$, w razie potrzeby dodając indeksy.

### 1.4.1. Typy proste

Mamy dany zbiór zmiennych typowych $TV$. Zbiór *typów prostych* to najmniejszy zbiór $STypes$ taki, że:

- jeśli $p \in TV$, to $p \in STypes$,
- jeśli $\tau$, $\sigma \in STypes$, to $(\tau \to \sigma) \in STypes$.

Zapisując typy proste opuszczamy zewnętrzne nawiasy. Przyjmujemy konwencję, że spójnik $\to$ wiąże w prawo, to znaczy $\tau_1 \to \tau_2 \to \tau_3 = (\tau_1 \to (\tau_2 \to \tau_3))$.

Dla danego typu prostego $\tau$ definiuję zbiór ścieżek $Paths(\tau)$ w następujący sposób:

- $Paths(p) = \{\epsilon\}$, jeśli $p$ jest zmienną typową,
- $Paths(\sigma_1 \to \sigma_2) = \{\epsilon\} \cup 1 \cdot Paths(\sigma_1) \cup 2 \cdot Paths(\sigma_2)$.

Elementy zbioru $Paths(\tau)$ będę nazywać *pozycjami*. Pozycje będę oznaczać literą $\rho$, być może uzupełnioną o indeks.

Będę mówić, że na pozycji $\rho$ w typie $\tau$ występuje zmienna typowa $p$, jeżeli:

- $\rho = \epsilon$ i $\tau = p$,
- $\rho = 1\rho_1$, $\tau = \tau_1 \to \tau_2$ i $p$ występuje na pozycji $\rho_1$ w typie $\tau_1$,
- $\rho = 2\rho_1$, $\tau = \tau_1 \to \tau_2$ i $p$ występuje na pozycji $\rho_1$ w typie $\tau_2$.

### 1.4.2. Typy pierwszego rzędu

Niech $\Sigma$ oznacza sygnaturę, czyli zbiór symboli relacyjnych i funkcyjnych o określonej arności. Symbole relacyjne będę oznaczać literami $P$, $Q$, $R$; symbole funkcyjne będę oznaczać literami $f$, $g$, $h$. Niech $Var$ oznacza pewien ustalony zbiór zmiennych algebraicznych. Zmienne algebraiczne będę oznaczać literami $\alpha$, $\beta$, $\gamma$. Zbiór *termów algebraicznych* o zmiennych $Var$ nad sygnaturą $\Sigma$ będę oznaczać $T(Var, \Sigma)$:

- jeśli $\alpha \in Var$, to $\alpha \in T(Var, \Sigma)$,

- jeśli $f \in \Sigma$ jest $n$-argumentowym symbolem funkcyjnym, $t_1$, ..., $t_n \in T(Var, \Sigma)$, to $f(t_1, ..., t_n) \in T(Var, \Sigma)$.

Termy algebraiczne będę oznaczać literami $t$, $s$.

Zbiór *typów pierwszego rzędu* nad ustaloną sygnaturą $\Sigma$ i ustalonym zbiorem zmiennych $Var$ będę oznaczać przez $Types_{Var,\Sigma}$:

- jeśli $P \in \Sigma$ jest $n$-argumentowym symbolem relacyjnym, $t_1$, ..., $t_n \in T(Var, \Sigma)$, to $P(t_1, ..., t_n) \in Types_{Var,\Sigma}$,
- jeśli $\tau$, $\sigma \in Types_{Var,\Sigma}$, to $(\tau \to \sigma) \in Types_{Var,\Sigma}$,
- jeśli $\tau \in Types_{Var,\Sigma}$, $\alpha \in Var$, to $\forall\alpha.\tau \in Types_{Var,\Sigma}$.

Podobnie jak dla typów prostych, opuszczamy zewnętrzne nawiasy i przyjmujemy konwencję, że spójnik $\to$ wiąże w prawo.

Symbol $FVar(\tau)$ oznacza algebraiczne zmienne wolne typu $\tau$, $BVar(\tau)$ oznacza zbiór algebraicznych zmiennych związanych typu $\tau$. Będziemy używać takich typów, w których każda zmienna jest związana co najwyżej jeden raz. Przez $Var(\tau)$ będę oznaczać zbiór zmiennych algebraicznych typu $\tau$. Analogicznie $FVar(t)$ oznacza zbiór zmiennych algebraicznych występujących w termie $t$.

Podobnie jak dla typów prostych, dla typów pierwszego rzędu definiujemy zbiór ścieżek $Paths(\tau)$:

- $Paths(P(t_1, ..., t_n)) = \{\epsilon\}$, jeśli $P$ jest $n$-argumentowym symbolem relacyjnym, a $t_1$, ..., $t_n$ są termami,
- $Paths(\sigma_1 \to \sigma_2) = \{\epsilon\} \cup 1 \cdot Paths(\sigma_1) \cup 2 \cdot Paths(\sigma_2)$,
- $Paths(\forall\alpha.\tau) = Paths(\tau)$.

*Arnością* formuły (typu pierwszego rzędu) $\tau$ będę nazywać liczbę $N_\tau$ taką, że

- jeśli $\tau = P(t_1, ..., t_n)$, to $N_\tau = n$,
- jeśli $\tau = \tau_1 \to \tau_2$, to $N_\tau = N_{\tau_1} + N_{\tau_2}$,
- jeśli $\tau = \forall\alpha.\tau_1$, to $N_\tau = N_{\tau_1}$.

Jeżeli $\alpha \notin FVar(\tau)$, to kwantyfikator $\forall\alpha$ występujący w formule $\forall\alpha.\tau$ nazwiemy *kwantyfikatorem niewiążącym*.

## 1.5. Podstawienia

Będziemy mieć do czynienia z dwoma rodzajami podstawień:

1. podstawienia algebraiczne, tj. funkcje ze zbioru $Var$ w zbiór $T(Var, \Sigma)$,
2. podstawienia typowe, tj. funkcje ze zbioru $TV$ w zbiór $STypes$.

W zależności od kontekstu będzie jasne, o jakim podstawieniu mowa. Podstawienia będę oznaczać literami $S$, $R$, ewentualnie z indeksami.

### 1.5.1. Podstawienia algebraiczne

Podstawienia algebraiczne rozszerzają się w oczywisty sposób na zbiór wszystkich typów pierwszego rzędu.

*Dziedziną* podstawienia nazywamy zbiór tych zmiennych, dla których podstawienie nie jest identycznością. Dziedzinę podstawienia $S$ oznaczamy przez $Dom(S)$. Przez $Ran(S)$ będziemy oznaczać zbiór $\bigcup_{\alpha \in Dom(\alpha)} FVar(S(\alpha))$.

Niech $\alpha$ będzie zmienną algebraiczną i niech $t$ będzie termem. Przez $S^t_\alpha$ będziemy oznaczać podstawienie takie, że $S^t_\alpha(\alpha) = t$ oraz $S^t_\alpha(\beta) = S(\beta)$ dla $\beta \neq \alpha$. Przez $[\alpha := t]$ będę oznaczać podstawienie, które zmiennej $\alpha$ przypisuje term $t$, a na pozostałych zmiennych jest identycznością.

### 1.5.2. Podstawienia typowe

Podstawienia typowe w oczywisty sposób rozszerzają się na zbiór wszystkich typów prostych.

*Dziedziną* podstawienia nazywamy zbiór tych zmiennych, dla których podstawienie nie jest identycznością. Dziedzinę podstawienia $S$ oznaczamy przez $Dom(S)$.

## 1.6. Środowisko

Środowisko to funkcja częściowa o skończonej dziedzinie ze zmiennych termowych $V$ w zbiór typów. W zależności od kontekstu będzie jasne, czy jest to funkcja w typy proste czy typy pierwszego rzędu. Środowiska będziemy oznaczać literami $\Gamma$, $\Delta$. Dziedzinę środowiska będziemy oznaczać przez $Dom(\Gamma)$. Przez $FVar(\Gamma)$ będziemy oznaczać zbiór $\bigcup_{x \in Dom(\Gamma)} FVar(\Gamma(x))$. Analogicznie definiujemy zbiór $BVar(\Gamma)$. Zakładamy, że wszystkie zmienne związane występujące w formułach w danym środowisku $\Gamma$ są różne. Ponadto $Var(\Gamma) = FVar(\Gamma) \cup BVar(\Gamma)$.

Jeśli $S$ jest podstawieniem algebraicznym, to przez $S(\Gamma)$ będziemy oznaczać środowisko takie, że:

- $Dom(S(\Gamma)) = Dom(\Gamma)$,
- $(S(\Gamma))(x) = S(\Gamma(x))$ dla każdego $x \in Dom(\Gamma)$.

## 1.7. Wyprowadzenia

Wyrażenie $\Gamma \vdash M : \tau$, gdzie $\Gamma$ jest środowiskiem, $M$ jest lambda termem, a $\tau$ jest typem będę nazywać *sekwentem*. Sekwenty będę oznaczać literą $S$. Typ $\tau$ występujący w sekwencie $S = \Gamma \vdash M : \tau$ będę nazywać *głównym typem* tego sekwentu i będę oznaczać $Type(S)$.

Wyprowadzenie składa się z sekwentu $S$, zera lub większej liczby podwyprowadzeń $D_1, ..., D_n$ oraz reguły $r$ takiej, że $S$ jest zastosowaniem reguły $r$ do konkluzji wyprowadzeń $D_1, ..., D_n$. Piszemy wówczas $\dfrac{D_1, ..., D_n}{S}\, r$. Wyprowadzenia będę oznaczać literą $D$. Sekwent $S$ w wyprowadzeniu $D = \dfrac{D_1, ..., D_n}{S}\, r$ będę oznaczać $\mathcal{FS}_D$ i będę nazywać *konkluzją* wyprowadzenia $D$.

Rozważamy system, w którym termom przypisujemy typy pierwszego rzędu, a reguły przypisywania typów są jak na rysunku 1.1. System ten będziemy nazywać *systemem typów pierwszego rzędu* lub *systemem pierwszego rzędu*. Wyprowadzenia w tym systemie będziemy często nazywać *wyprowadzeniami pierwszego rzędu*. *System typów prostych* to system, w którym występują tylko typy proste oraz jedynymi regułami są (Ax), (App) i (Abs).

$$\text{(Ax)}\ \Gamma, x : \tau \vdash x : \tau$$

$$\text{(Abs)}\ \frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma} \qquad \text{(App)}\ \frac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma}$$

$$\text{(Gen)}\ \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M : \forall \alpha.\sigma}\ \alpha \notin FVar(\Gamma) \qquad \text{(Inst)}\ \frac{\Gamma \vdash M : \forall \alpha.\sigma}{\Gamma \vdash M : \sigma[\alpha := t]}$$

Rysunek 1.1: Reguły przypisywania typów w systemie typów pierwszego rzędu

Niech $D$ będzie wyprowadzeniem pierwszego rzędu, $D = \dfrac{D_1, ..., D_n}{\Gamma \vdash M : \tau}\ r$. Definiujemy zbiór zmiennych związanych wyprowadzenia $D$ jako

$$BVar(D) = BVar(\tau) \cup \bigcup_{x \in Dom(\Gamma)} BVar(\Gamma(x)) \cup \bigcup_{i \leqslant n} BVar(D_i).$$

Analogicznie definiujemy zbiór zmiennych wolnych $FVar(D)$ oraz zbiór wszystkich zmiennych $Var(D)$.

Niech $D$ będzie poprawnym wyprowadzeniem o konkluzji $\Gamma \vdash M : \tau$. Niech $x$ będzie zmienną termu $M$. Przez $IDT_D(x)$ będę oznaczać typ przypisany w wyprowadzeniu $D$ zmiennej $x$ za pomocą reguły (Ax). To jest dobrze zdefiniowane pojęcie, bo dla zmiennej wolnej $IDT_D(x) = \Gamma(x)$. Ponadto każda zmienna związana $x$ jest związana w $M$ dokładnie jeden raz, więc $IDT_D(x) = \Delta(x)$, gdzie $\Delta$ jest środowiskiem pojawiającym się w $D$ takim, że $x \in Dom(\Delta)$.

**Definicja 1.7.1.** *Problem sprawdzania typu*, to następujący problem: mając dany term $M$, typ $\tau$ i środowisko $\Gamma$ rozstrzygnąć, czy istnieje poprawne wyprowadzenie, w którym konkluzją jest $\Gamma \vdash M : \tau$.

*Warunkiem brzegowym* dla problemu sprawdzania typu, będę nazywać sekwent $\Gamma \vdash M : \tau$.

# Rozdział 2

# Problem sprawdzania typu

Celem tej pracy jest zbadanie, czy problem sprawdzania typu dla systemu typów pierwszego rzędu jest rozstrzygalny. Chcemy stwierdzić, czy istnieje procedura, która mając dany term $M$, typ pierwszego rzędu $\tau$ i środowisko $\Gamma$ odpowiada, czy $\Gamma \vdash M : \tau$ jest wyprowadzalne. Zakładamy ponadto, że w wyprowadzeniu można używać tylko tych symboli funkcyjnych i relacyjnych, które występują w warunku brzegowym.

Zobaczmy następujący przykład.

**Przykład 2.0.2.** Niech $\Gamma$ oznacza $y : \forall\alpha\forall\beta.P(\alpha,\beta), x : \forall\gamma P(\gamma, f(\gamma)) \to Q$. Mamy następujące wyprowadzenie.

$$\dfrac{\dfrac{\Gamma \vdash x : \forall\gamma P(\gamma, f(\gamma)) \to Q \qquad \dfrac{\dfrac{\dfrac{\Gamma \vdash y : \forall\alpha\forall\beta.P(\alpha,\beta)}{\Gamma \vdash y : \forall\beta.P(\delta,\beta)}}{\Gamma \vdash y : P(\delta, f(\delta))}}{\Gamma \vdash y : \forall\delta.P(\delta, f(\delta))}}{\Gamma \vdash xy : Q}}{y : \forall\alpha\forall\beta.P(\alpha,\beta) \vdash \lambda x.xy : (\forall\gamma P(\gamma, f(\gamma)) \to Q) \to Q}$$

Jak widać reguły (Gen) i (Inst) były potrzebne, by wyprowadzić konkluzję. Jest jasne, że nie można skonstruować wyprowadzenia o konkluzji

$$y : \forall\alpha\forall\beta.P(\alpha,\beta) \vdash \lambda x.xy : (\forall\gamma P(\gamma, f(\gamma)) \to Q) \to Q,$$

w którym nie występują reguły (Gen) oraz (Inst).

Dowiedziemy, że problem sprawdzania typu dla systemu typów pierwszego rzędu w pewnej ograniczonej wersji jest rozstrzygalny.

## 2.1. System typów prostych a system typów pierwszego rzędu

System typów pierwszego rzędu jest rozszerzeniem systemu typów prostych. Naturalne jest zatem badanie związków pomiędzy tymi systemami. Zajmiemy się teraz tym zagadnieniem. Jak się okaże, na wyprowadzenie pierwszego rzędu można patrzeć jak na wyprowadzenie w typach prostych wzbogacone o pewne nowe elementy.

Na początek zdefiniujemy translację z systemu typów pierwszego rzędu do systemu typów prostych. W tym celu będzie potrzebnych kilka dodatkowych konstrukcji.

Definiujemy operator $ST : Types \to STypes$ w następujący sposób:

- $ST(P(t_1, ..., t_n)) = P$,
- $ST(\sigma \to \tau) = ST(\sigma) \to ST(\tau)$,
- $ST(\forall\alpha.\tau) = ST(\tau)$.

Możemy rozszerzyć działanie operatora $ST$ na środowiska. Dla danego $\Gamma$, $ST(\Gamma)$ to środowisko takie, że:

- $Dom(ST(\Gamma)) = Dom(\Gamma)$,
- $(ST(\Gamma))(x) = ST(\Gamma(x))$ dla $x \in Dom(\Gamma)$.

Następnie rozszerzamy operację $ST$ na wszystkie sekwenty wyprowadzenia w następujący sposób: $ST(\Gamma \vdash M : \tau) = ST(\Gamma) \vdash M : ST(\tau)$.

**Definicja 2.1.1.** $ST(D)$ będzie oznaczać wyprowadzenie, które powstaje przez zaaplikowanie operacji $ST$ do każdego sekwentu wyprowadzenia $D$ oraz usunięcie fragmentów postaci $\frac{S}{S}$.

**Lemat 2.1.2.** *Jeżeli $D$ jest poprawnym wyprowadzeniem w logice pierwszego rzędu, to $ST(D)$ jest poprawnym wyprowadzeniem w typach prostych.*

*Dowód.* Łatwa indukcja ze względu na rozmiary wyprowadzenia $D$. □

Otrzymujemy następujący wniosek.

**Wniosek 2.1.3.** *Jeżeli $\Gamma \vdash M : \tau$ w typach pierwszego rzędu, to $ST(\Gamma) \vdash M : ST(\tau)$ w typach prostych.*

Zatem warunkiem koniecznym na to, żeby zachodziło $\Gamma \vdash M : \tau$ w typach pierwszego rzędu jest $ST(\Gamma) \vdash M : ST(\tau)$ w typach prostych. Ujmując to nieco inaczej, na wyprowadzenie pierwszego rzędu można patrzeć jak na pewne udekorowanie wyprowadzenia w typach prostych. Aby otrzymać poprawne wyprowadzenie pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$, należy wzbogacić wyprowadzenie o konkluzji $ST(\Gamma \vdash M : \tau)$ poprzez dodanie w niektórych miejscach reguł (Inst) lub (Gen) oraz odpowiednie dopisanie kwantyfikatorów i termów algebraicznych w typach w tym wyprowadzeniu. Zauważmy, że nie zawsze jest to możliwe.

**Przykład 2.1.4.** Sekwent $S_1 = x : (p \to q) \to r, y : p \to q \vdash xy : r$ jest wyprowadzalny w typach prostych. Mamy bowiem następujące wyprowadzenie.

$$\frac{x : (p \to q) \to r, y : p \to q \vdash x : (p \to q) \to r \qquad x : (p \to q) \to r, y : p \to q \vdash y : p \to q}{x : (p \to q) \to r, y : p \to q \vdash xy : r}$$

Weźmy sekwent $S_2 = x : (P(\beta) \to \forall\alpha.Q(\alpha, \beta)) \to R, y : P(\beta) \to \forall\gamma.Q(\beta, \gamma) \vdash xy : R$. Zauważmy, że $ST(S_2) = S_1$, ale $S_2$ nie jest wyprowadzalny w logice pierwszego rzędu. Gdyby dało się go wyprowadzić, to ostatnią regułą musiałaby być reguła (App). To oznacza, że trzeba by uzgodnić formuły $P(\beta) \to \forall\alpha.Q(\alpha, \beta)$ i $P(\beta) \to \forall\gamma.Q(\beta, \gamma)$, to zaś jest niemożliwe.

W dalszej części pracy zajmiemy się dokładniejszym zbadaniem, w jaki sposób można wzbogacać strukturę wyprowadzenia w typach prostych i otrzymać poprawne wyprowadzenie pierwszego rzędu.

Wiadomo (np. [3], [5]), że dla każdego termu $M$ typowalnego w typach prostych istnieje typ główny $\widehat{\tau}_M$ i wyprowadzenie główne $\widehat{D}_M$. Wyprowadzenie główne i typ główny są wyznaczone jednoznacznie z dokładnością do wyboru nazw zmiennych typowych.

**Definicja 2.1.5.** Zmienne typowe, które występują w konkluzji wyprowadzenia $\widehat{D}_M$ będę nazywać *starymi* zmiennymi wyprowadzenia $D$. Pozostałe zmienne typowe pojawiające się w wyprowadzeniu $D$ będę nazywać *nowymi* zmiennymi wyprowadzenia $D$.

Wiadomo, że każde wyprowadzenie w typach prostych jest instancją wyprowadzenia głównego. W szczególności, jeżeli $D$ jest wyprowadzeniem pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$, to $ST(D)$ jest instancją wyprowadzenia $\widehat{D}_M$. Istnieje podstawienie typowe $R$ takie, że wyprowadzenie $ST(D)$ jest rezultatem aplikacji $R$ do wyprowadzenia głównego $\widehat{D}_M$. Mając dany warunek brzegowy $\Gamma \vdash M : \tau$ i wyprowadzenie główne $\widehat{D}_M$, możemy częściowo odtworzyć podstawienie $R$. Porównując $\Gamma \vdash M : \tau$ i $\mathcal{FS}_{\widehat{D}_M}$, zobaczymy jak jest zdefiniowane $R$ na starych zmiennych typowych $\widehat{D}_M$. Nie wiadomo jednak nic o tym, jakie wartości przyjmuje $R$ na nowych zmiennych typowych. Okazuje się jednak, że możemy przyjąć pewne założenia co do ich postaci.

**Definicja 2.1.6.** Wyprowadzenie $E$ o konkluzji $\Gamma \vdash M : \tau$ w typach prostych nazwiemy *prostym*, jeżeli jest ono wynikiem aplikacji do $\widehat{D}_M$ podstawienia typowego $R$ takiego, że dla każdej nowej zmiennej typowej $q$ formuła $R(q)$ jest zmienną typową.

Pokażemy, że można założyć, iż wyprowadzenie $ST(D)$ jest wyprowadzeniem prostym.

Jak już zostało powiedziane, $ST(D)$ jest pewną instancją wyprowadzenia głównego. Jest jasne, że każdemu sekwentowi wyprowadzenia $ST(D)$ odpowiada pewien sekwent wyprowadzenia głównego $\widehat{D}_M$. Jednakże również każdemu sekwentowi $S$ wyprowadzenia $D$ odpowiada pewien sekwent w wyprowadzeniu głównym $\widehat{D}_M$. Ten odpowiadający $S$ sekwent w wyprowadzeniu głównym nazwiemy *wzorcem* dla $S$. Następująca definicja formalizuje pojęcie wzorca:

**Definicja 2.1.7.** Dla każdego sekwentu $S$ wyprowadzenia $D$ definiujemy *wzorzec* $P_D(S)$:

- $P_D(\mathcal{FS}_D) = \mathcal{FS}_{\widehat{D}_M}$.
- Jeśli $S$ jest konkluzją reguły (Abs) $\dfrac{S_1}{S}$ i $P_D(S) = \widehat{S}$, to $\widehat{S}$ jest konkluzją reguły (Abs) postaci $\dfrac{\widehat{S_1}}{\widehat{S}}$. Wtedy $P_D(S_1) := \widehat{S_1}$.
- Jeśli $S$ jest konkluzją reguły (App) $\dfrac{S_1 \qquad S_2}{S}$ i $P_D(S) = \widehat{S}$, to $\widehat{S}$ jest konkluzją reguły (Abs) postaci $\dfrac{\widehat{S_1} \qquad \widehat{S_2}}{\widehat{S}}$. Wtedy $P_D(S_1) := \widehat{S_1}$ i $P_D(S_2) := \widehat{S_2}$.
- Jeśli $S$ jest konkluzją reguły (Gen) $\dfrac{S_1}{S}$ i $P_D(S) = \widehat{S}$, to $P_D(S_1) := \widehat{S}$.
- Jeśli $S$ jest konkluzją reguły (Inst) $\dfrac{S_1}{S}$ i $P_D(S) = \widehat{S}$, to $P_D(S_1) := \widehat{S}$.

Łatwo można zrobić następującą obserwację.

**Uwaga 2.1.8.** Dla każdej instancji reguły (Gen) lub (Inst): $\dfrac{S_1}{S_2}$ zachodzi $P_D(S_1) = P_D(S_2)$.

Pokazujemy, że można założyć, że wyprowadzenie $ST(D)$ jest proste.

Niech $Q$ będzie dowolnym ustalonym zeroargumentowym symbolem relacyjnym. Niech $M$ będzie lambda termem. Intensjonalne znaczenie poniższej operacji jest takie: dana jest formuła $\sigma$ oraz odpowiadający jej wzorzec (formuła rachunku zdań) $\tau$, $Clear_M(\sigma, \tau)$ to taka formuła, w której starym zmiennym typowym odpowiadają te same formuły, co w $\sigma$, a nowym zmiennym typowym ustalona formuła $Q$. Definiujemy operację $Clear_M : Types \times STypes \to Types$:

- $Clear_M(\sigma, p) = \sigma$, jeśli $p$ jest starą zmienną typową wyprowadzenia $\widehat{D}_M$,
- $Clear_M(\sigma, q) = Q$, jeśli $q$ jest nową zmienną typową wyprowadzenia $\widehat{D}_M$,
- $Clear_M(\forall\alpha.\sigma, \tau) = \forall\alpha.Clear_M(\sigma, \tau)$, jeśli $\tau$ nie jest zmienną typową,
- $Clear_M(\sigma_1 \to \sigma_2, \tau_1 \to \tau_2) = Clear_M(\sigma_1, \tau_1) \to Clear_M(\sigma_2, \tau_2)$.

**Lemat 2.1.9.** *Dla dowolnego typu $\sigma \in Types$ i $\widehat{\sigma} \in STypes$ takiego, że $Clear_M(\sigma, \widehat{\sigma})$ jest określone, zachodzi $Clear_M(\sigma[\alpha := t], \widehat{\sigma}) = Clear_M(\sigma, \widehat{\sigma})[\alpha := t]$.*

*Dowód.* Indukcja ze względu na definicję operacji $Clear_M$.

- $Clear_M(\sigma[\alpha := t], p) = \sigma[\alpha := t] = Clear_M(\sigma, p)[\alpha := t]$, jeśli $p$ jest starą zmienną typową wyprowadzenia $\widehat{D}_M$.
- $Clear_M(\sigma[\alpha := t], q) = Q = Q[\alpha := t] = Clear_M(\sigma, q)[\alpha := t]$, jeśli $q$ jest nową zmienną typową wyprowadzenia $\widehat{D}_M$.
- Jeśli $\beta \neq \alpha$, to $Clear_M((\forall\beta.\sigma)[\alpha := t], \tau) = Clear_M(\forall\beta.(\sigma[\alpha := t]), \tau) = = \forall\beta.Clear_M(\sigma[\alpha := t], \tau) = \forall\beta.Clear_M(\sigma, \tau)[\alpha := t] = (\forall\beta.Clear_M(\sigma, \tau))[\alpha := t]$.
- Jeśli $\beta = \alpha$, to $Clear_M((\forall\beta.\sigma)[\alpha := t], \tau) = Clear_M(\forall\beta.\sigma, \tau) = \forall\beta.Clear_M(\sigma, \tau) = = (\forall\beta.Clear_M(\sigma, \tau))[\alpha := t]$.
- $Clear_M((\sigma_1 \to \sigma_2)[\alpha := t], \tau_1 \to \tau_2) = Clear_M(\sigma_1[\alpha := t] \to \sigma_2[\alpha := t], \tau_1 \to \tau_2) = Clear_M(\sigma_1[\alpha := t], \tau_1) \to Clear_M(\sigma_2[\alpha := t], \tau_2) = Clear_M(\sigma_1, \tau_1)[\alpha := t] \to Clear_M(\sigma_2, \tau_2)[\alpha := t] = (Clear_M(\sigma_1, \tau_1) \to Clear_M(\sigma_2, \tau_2))[\alpha := t]$.

□

Niech $D$ będzie wyprowadzeniem pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$. Używając operacji $Clear_M$ skonstruujemy nowe wyprowadzenie pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$.

Najpierw definiujemy środowisko $[\Gamma]$:

- $Dom([\Gamma]) = Dom(\Gamma)$,
- $[\Gamma](x) = Clear_M(\Gamma(x), IDT_{\widehat{D}_M}(x))$.

Następnie definiujemy działanie operacji [ . ] na sekwentach wyprowadzenia $D$. Dla każdego sekwentu $S$ postaci $\Gamma \vdash N : \sigma$ określamy $[S] = [\Gamma] \vdash N : Clear_M(\sigma, Type(P_D(S)))$.

**Definicja 2.1.10.** Wyprowadzenie $[D]$ jest wyprowadzeniem, które powstaje z wyprowadzenia $D$ poprzez zastosowanie operacji [ . ] do każdego jego sekwentu oraz usunięcie podwyprowadzeń postaci $\dfrac{S}{S}$.

**Lemat 2.1.11.** *Niech $D$ będzie wyprowadzeniem w typach pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$. Wtedy $[D]$ jest poprawnym wyprowadzeniem pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$.*

*Dowód.* Z konstrukcji jest jasne, że $\mathcal{FS}_{[D]} = \mathcal{FS}_D$.

Pokażemy, że zastosowanie operacji [ . ] do kolejnych sekwentów wyprowadzenia $D$ prowadzi albo do poprawnego użycia jednej z reguł, albo do podwyprowadzenia $\dfrac{S}{S}$. Sprawdzamy kolejne przypadki:

- Sekwent $S$ w $D$ jest zastosowaniem reguły (Ax) postaci $\Gamma, x : \tau \vdash x : \tau$. Niech $P_D(S) = \Delta, x : \widehat{\tau} \vdash x : \widehat{\tau}$. Wtedy $IDT_{\widehat{D}_M(x)} = \widehat{\tau}$. Czyli

$$[S] = [\Gamma], x : Clear_M(\tau, \widehat{\tau}) \vdash x : Clear_M(\tau, \widehat{\tau}),$$

  a więc $[S]$ jest poprawnym zastosowaniem reguły (Ax).

- Zastosowano regułę (App) postaci $\dfrac{S_1 \qquad S_2}{S}$, gdzie

$$\begin{aligned} S_1 &= \Gamma \vdash N : \sigma_1 \to \sigma_2, \\ S_2 &= \Gamma \vdash L : \sigma_1, \\ S &= \Gamma \vdash NL : \sigma_2. \end{aligned}$$

  Niech $P_D(S_2) = \Delta \vdash L : \widehat{\sigma}_1$, $P_D(S) = \Delta \vdash NL : \widehat{\sigma}_2$. Wtedy $P_D(S_1)$ musi być postaci $\Delta \vdash N : \widehat{\sigma}_1 \to \widehat{\sigma}_2$, bo $\widehat{D}_M$ jest poprawnym wyprowadzeniem. Wówczas jednak

$$\begin{aligned} [S_1] &= [\Gamma] \vdash N : Clear_M(\sigma_1 \to \sigma_2, \widehat{\sigma}_1 \to \widehat{\sigma}_2), \\ [S_2] &= [\Gamma] \vdash L : Clear_M(\sigma_1, \widehat{\sigma}_1), \\ [S] &= [\Gamma] \vdash NL : Clear_M(\sigma_2, \widehat{\sigma}_2). \end{aligned}$$

  Ale $Clear_M(\sigma_1 \to \sigma_2, \widehat{\sigma}_1 \to \widehat{\sigma}_2) = Clear_M(\sigma_1, \widehat{\sigma}_1) \to Clear_M(\sigma_2, \widehat{\sigma}_2)$ i jest dobrze.

- Zastosowano regułę (Abs) postaci $\dfrac{S_1}{S_2}$, gdzie

$$\begin{aligned} S_1 &= \Gamma, x : \sigma_1 \vdash N : \sigma_2, \\ S_2 &= \Gamma \vdash \lambda x.N : \sigma_1 \to \sigma_2. \end{aligned}$$

  Wówczas

$$\begin{aligned} P_D(S_1) &= \Delta, x : \widehat{\sigma}_1 \vdash N : \widehat{\sigma}_2, \\ P_D(S_2) &= \Delta \vdash \lambda x.N : \widehat{\sigma}_1 \to \widehat{\sigma}_2. \end{aligned}$$

  Wtedy

$$\begin{aligned} [S_1] &= [\Gamma], x : Clear_M(\sigma_1, \widehat{\sigma}_1) \vdash N : Clear_M(\sigma_2, \widehat{\sigma}_2), \\ [S_2] &= [\Gamma] \vdash \lambda x.M : Clear_M(\sigma_1 \to \sigma_2, \widehat{\sigma}_1 \to \widehat{\sigma}_2). \end{aligned}$$

  Jednak $Clear_M(\sigma_1 \to \sigma_2, \widehat{\sigma}_1 \to \widehat{\sigma}_2) = Clear_M(\sigma_1, \widehat{\sigma}_1) \to Clear_M(\sigma_2, \widehat{\sigma}_2)$, czyli jest dobrze.

- Zastosowano regułę (Gen) postaci $\dfrac{S_1}{S}$, gdzie $S_1 = \Gamma \vdash N : \sigma$, $S = \Gamma \vdash N : \forall\alpha.\sigma$. Wtedy zachodzi: $P_D(S_1) = P_D(S) = \Delta \vdash N : \widehat{\sigma}$. Wówczas

$$\begin{aligned} [S_1] &= [\Gamma] \vdash N : Clear_M(\sigma, \widehat{\sigma}), \\ [S] &= [\Gamma] \vdash N : Clear_M(\forall\alpha.\sigma, \widehat{\sigma}). \end{aligned}$$

  Jeśli $\widehat{\sigma}$ jest nową zmienną typową, to $Clear_M(\sigma, \widehat{\sigma}) = Clear_M(\forall\alpha.\sigma, \widehat{\sigma}) = Q$, czyli $[S] = [S_1]$. Otrzymujemy dwa takie same sekwenty.

  Jeżeli $\widehat{\sigma}$ nie jest nową zmienną typową, to $Clear_M(\forall\alpha.\sigma, \widehat{\sigma}) = \forall\alpha.Clear_M(\sigma, \widehat{\sigma})$. Oczywiście skoro $\alpha \notin FVar(\Gamma)$, to $\alpha \notin FVar([\Gamma])$, czyli mamy poprawne użycie reguły (Gen).

- Zastosowano regułę (Inst) postaci $\frac{S_1}{S}$, gdzie

$$\begin{aligned} S_1 &= \Gamma \vdash N : \forall\alpha.\sigma, \\ S &= \Gamma \vdash N : \sigma[\alpha := t]. \end{aligned}$$

  Wtedy $P_D(S_1) = P_D(S) = \Delta \vdash N : \widehat{\sigma}$. Wówczas $[S_1] = [\Gamma] \vdash N : Clear_M(\forall\alpha.\sigma, \widehat{\sigma})$, $[S] = [\Gamma] \vdash N : Clear_M(\sigma[\alpha := t], \widehat{\sigma})$.

  Jeżeli $\widehat{\sigma}$ jest nową zmienną typową, to $Clear_M(\forall\alpha.\sigma, \widehat{\sigma}) = Clear_M(\sigma[\alpha := t], \widehat{\sigma}) = Q$ i jest dobrze.

  W przeciwnym przypadku $Clear_M(\forall\alpha.\sigma, \widehat{\sigma}) = \forall\alpha.Clear_M(\sigma, \widehat{\sigma})$ oraz z lematu 2.1.9 zachodzi $Clear_M(\sigma[\alpha := t], \widehat{\sigma}) = Clear_M(\sigma, \widehat{\sigma})[\alpha := t]$, czyli mamy poprawne użycie reguły (Inst).

□

**Uwaga 2.1.12.** Powyższa konstrukcja nie zależy od tego, czy w sygnaturze jest zeroargumentowy symbol relacyjny $Q$. Jeżeli takiego symbolu nie ma, możemy $Q$ zastąpić formułą $\forall\alpha.P(\alpha, ..., \alpha)$ dla pewnego symbolu relacyjnego $P$ z sygnatury. Łatwo sprawdzić, że powyższe rozumowania są poprawne także w tym przypadku.

Otrzymujemy zatem następujący wniosek.

**Wniosek 2.1.13.** *Jeżeli istnieje wyprowadzenie $D$ w typach pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$, to istnieje również wyprowadzenie $D'$ o takiej samej konkluzji i prostym $ST(D')$.*

## 2.2. Wyprowadzenia typu INST-before-GEN

Jak stwierdziliśmy poprzednio, wyprowadzenie pierwszego rzędu w pewnym sensie jest wyprowadzeniem w typach prostych, wzbogaconym o odpowiednie rozmieszczenie reguł (Inst) lub (Gen) oraz kwantyfikatorów i termów algebraicznych. Zbadamy, jakie ograniczenia można narzucić na rozmieszczenie reguł (Inst) lub (Gen).

Wprowadźmy następującą definicję, zaczerpniętą z [7].

**Definicja 2.2.1.** Wyprowadzenie $D$ ma własność *INST-before-GEN* wtedy i tylko wtedy, gdy nie ma w nim ciągu sekwentów:

$$\frac{\dfrac{\Gamma \vdash M : \sigma}{\Gamma \vdash M : \forall\alpha.\sigma}\,(\text{Gen})}{\Gamma \vdash M : \sigma[\alpha := t]}\,(\text{Inst})$$

Pokażemy, że można rozważać tylko wyprowadzenia mające własność *INST-before-GEN*. Potrzebny nam będzie następujący lemat.

**Lemat 2.2.2.** *Niech $D$ będzie wyprowadzeniem, w którym konkluzją jest sekwent $\Gamma \vdash M : \tau$. Niech $S$ będzie podstawieniem algebraicznym, w którym $Ran(S) \cap BVar(D) = \emptyset$. Wówczas istnieje wyprowadzenie $D'$ mające taką samą strukturę reguł jak wyprowadzenie $D$, w którym konkluzją jest $S(\Gamma) \vdash M : S(\tau)$.*

*Dowód.* Indukcja ze względu na ostatnio zastosowaną regułę w wyprowadzeniu $D$.

- Jeśli ostatnią zastosowaną regułą był aksjomat $\Gamma, x : \tau \vdash x : \tau$, to oczywiście zachodzi $S(\Gamma), x : S(\tau) \vdash x : S(\tau)$.

- Jeżeli ostatnią użytą regułą była reguła (Abs):

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma},$$

to zachodzi

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

Ale $S(\tau) \to S(\sigma) = S(\tau \to \sigma)$ z definicji podstawienia, więc otrzymujemy tezę.

- Ostatnią regułą była reguła (App):

$$\frac{\Gamma \vdash N : \sigma \to \tau \qquad \Gamma \vdash L : \sigma}{\Gamma \vdash NL : \tau}.$$

Wtedy z założenia indukcyjnego $S(\Gamma) \vdash N : S(\sigma \to \tau)$ oraz $S(\Gamma) \vdash L : S(\sigma)$. Ale $S(\sigma \to \tau) = S(\sigma) \to S(\tau)$. Zatem

$$\frac{S(\Gamma) \vdash N : S(\sigma) \to S(\tau) \qquad S(\Gamma) \vdash L : S(\sigma)}{S(\Gamma) \vdash NL : S(\tau)}.$$

- Ostatnią regułą była generalizacja:

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M : \forall \alpha.\sigma}.$$

Wtedy $\alpha \notin FVar(\Gamma)$. Korzystając z założenia indukcyjnego dla podstawienia $S^\alpha_\alpha$ i wyprowadzenia $\Gamma \vdash M : \sigma$ otrzymamy:

$$S^\alpha_\alpha(\Gamma) \vdash M : S^\alpha_\alpha(\sigma).$$

Z założenia zachodzi $Ran(S) \cap BVar(D) = \emptyset$, więc $\alpha \notin FVar(S(\Gamma))$. Możemy zastosować regułę (Gen):

$$\frac{S^\alpha_\alpha(\Gamma) \vdash M : S^\alpha_\alpha(\sigma)}{S^\alpha_\alpha(\Gamma) \vdash M : \forall \alpha.S^\alpha_\alpha(\sigma)}.$$

Ale wtedy $\forall \alpha.S^\alpha_\alpha(\sigma) = S(\forall \alpha.\sigma)$ i $S^\alpha_\alpha(\Gamma) = S(\Gamma)$, czyli otrzymujemy tezę lematu.

- Ostatnią regułą jest (Inst):

$$\frac{\Gamma \vdash M : \forall \alpha.\tau}{\Gamma \vdash M : \tau[\alpha := t]}.$$

Wtedy z założenia indukcyjnego $S(\Gamma) \vdash M : S(\forall \alpha.\tau)$. Ale $S(\forall \alpha.\tau) = \forall \alpha.S^\alpha_\alpha(\tau)$. Czyli $S(\Gamma) \vdash M : \forall \alpha.S^\alpha_\alpha(\tau)$. Ponadto $S^\alpha_\alpha(\Gamma) = S(\Gamma)$, bo rozważamy tylko takie wyprowadzenia, w których $FVar(\Gamma) \cap BVar(D) = \emptyset$. Zatem

$$\frac{S^\alpha_\alpha(\Gamma) \vdash M : \forall\alpha.S^\alpha_\alpha(\tau)}{S^\alpha_\alpha(\Gamma) \vdash M : S^\alpha_\alpha(\tau)[\alpha := S(t)]}\ .$$

Jednak $S^\alpha_\alpha(\tau)[\alpha := S(t)] = S(\tau[\alpha := \tau])$. Zatem istotnie sekwent $S(\Gamma) \vdash M : S(\tau[\alpha{:=}t])$ jest wyprowadzalny.

□

**Lemat 2.2.3.** *Niech $D$ będzie wyprowadzeniem, w którym konkluzją jest sekwent $\Gamma \vdash M : \tau$. Wtedy istnieje wyprowadzenie $D'$ mające własność INST-before-GEN, w którym konkluzją jest sekwent $\Gamma \vdash M : \tau$.*

*Dowód.* Jeśli $D$ nie ma własności *INST-before-GEN*, to istnieje w nim podwyprowadzenie:

$$\frac{\dfrac{\Gamma \vdash M : \tau}{\Gamma \vdash M : \forall\alpha.\tau}}{\Gamma \vdash M : \tau[\alpha := t]}\ .$$

Możemy założyć, ze w $FVar(t) \cap BVar(\tau) = \emptyset$, w razie konieczności wymieniając zmienne związane na inne. Ponadto z założenia $\alpha$ nie jest zmienną wolną w żadnym typie pojawiającym się w $\Gamma$. Stosuję lemat 2.2.2 do wyprowadzenia $\Gamma \vdash M : \tau$ i podstawienia $[\alpha := t]$. Dostaję wyprowadzenie o konkluzji $\Gamma \vdash M : \tau[\alpha := t]$, o tej samej strukturze reguł jak wyprowadzenie $\Gamma \vdash M : \tau$ w $D$. Jest ono oczywiście mniejsze niż oryginalne wyprowadzenie. Tę konstrukcję możemy zastosować po kolei do każdego ciągu reguł postaci

$$\frac{\dfrac{\Gamma \vdash M : \tau}{\Gamma \vdash M : \forall\alpha.\tau}}{\Gamma \vdash M : \tau[\alpha := t]}\ .$$

W rezultacie otrzymamy wyprowadzenie $D'$ o konkluzji $\Gamma \vdash M : \tau$, mające własność *INST-before-GEN*. □

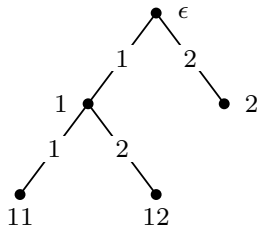
## 2.3. Kwantyfikatory niewiążące

Zajmiemy się teraz kwantyfikatorami niewiążącymi, które występują w wyprowadzeniu pierwszego rzędu. Intuicyjnie jest jasne, że nie niosą one żadnej istotnej informacji, gdyż zastosowanie reguły (Inst) do takiego kwantyfikatora właściwie nie zmienia typu. Nie możemy jednak zupełnie odrzucić kwantyfikatorów niewiążących, bo mogą one wystąpić w warunku brzegowym. W niektórych miejscach wystąpienie takiego kwantyfikatora jest wówczas nieuniknione. Pokażemy jednak, że można nie rozważać kwantyfikatorów niewiążących w miejscach, gdzie warunek brzegowy nie wymusza ich wystąpienia.

**Definicja 2.3.1.** *Grafem typu prostego $\tau$* będę nazywać nieskierowany graf $\mathcal{G}(\tau) = (V, E)$ o krawędziach etykietowanych 1, 2 taki, że

- $V = Paths(\tau)$,
- jeśli $\rho \in Paths(\tau)$ i $\rho 1 \in Paths(\tau)$, to $(\rho 1 \overset{1}{\longleftrightarrow} \rho) \in E$,
- jeśli $\rho \in Paths(\tau)$ i $\rho 2 \in Paths(\tau)$, to $(\rho 2 \overset{2}{\longleftrightarrow} \rho) \in E$.

Rysunek 2.1 przedstawia graf typu $(p \to q) \to p$.




Rysunek 2.1: Graf typu

**Definicja 2.3.2.** *Grafem typu pierwszego rzędu* $\tau$ będę nazywać graf odpowiadającego mu typu prostego $ST(\tau)$.

Ten sam rysunek 2.1 przedstawia graf typu $\forall\alpha(\forall\beta(P(\beta,\alpha) \to Q(\gamma)) \to P(\alpha, f(\alpha)))$.

W danym wyprowadzeniu w typach prostych $E$ o konkluzji $\Gamma \vdash M : \tau$, każdemu podtermowi $N \in SubT(M)$ jeden raz lub więcej razy zostaje przypisany typ. Łatwo jednak zauważyć, że jeżeli typ dla termu $N$ jest przypisany więcej niż jeden raz, to za każdym razem jest to ten sam typ. (Przypominam, że nie utożsamiamy alfa-równoważnych podtermów.) Analogicznie łatwo sprawdzić, że w wyprowadzeniu pierwszego rzędu $D$ term $N$ dostaje typy o takim samym grafie. Ten graf będę oznaczać przez $\mathcal{G}_D(N)$.

**Definicja 2.3.3.** *Prostym grafem typów* dla wyprowadzenia (w typach prostych lub typach pierwszego rzędu) $D$ będę nazywać sumę rozłączną grafów $(\mathcal{G}_D(N))_{N \in SubT(M)}$. Graf ten będę oznaczać przez $\mathcal{G}^0(D)$.

**Przykład 2.3.4.** Rozważmy następujące wyprowadzenie. Tutaj $\Gamma$ oznacza środowisko $x : (p \to q) \to r, u : p$.

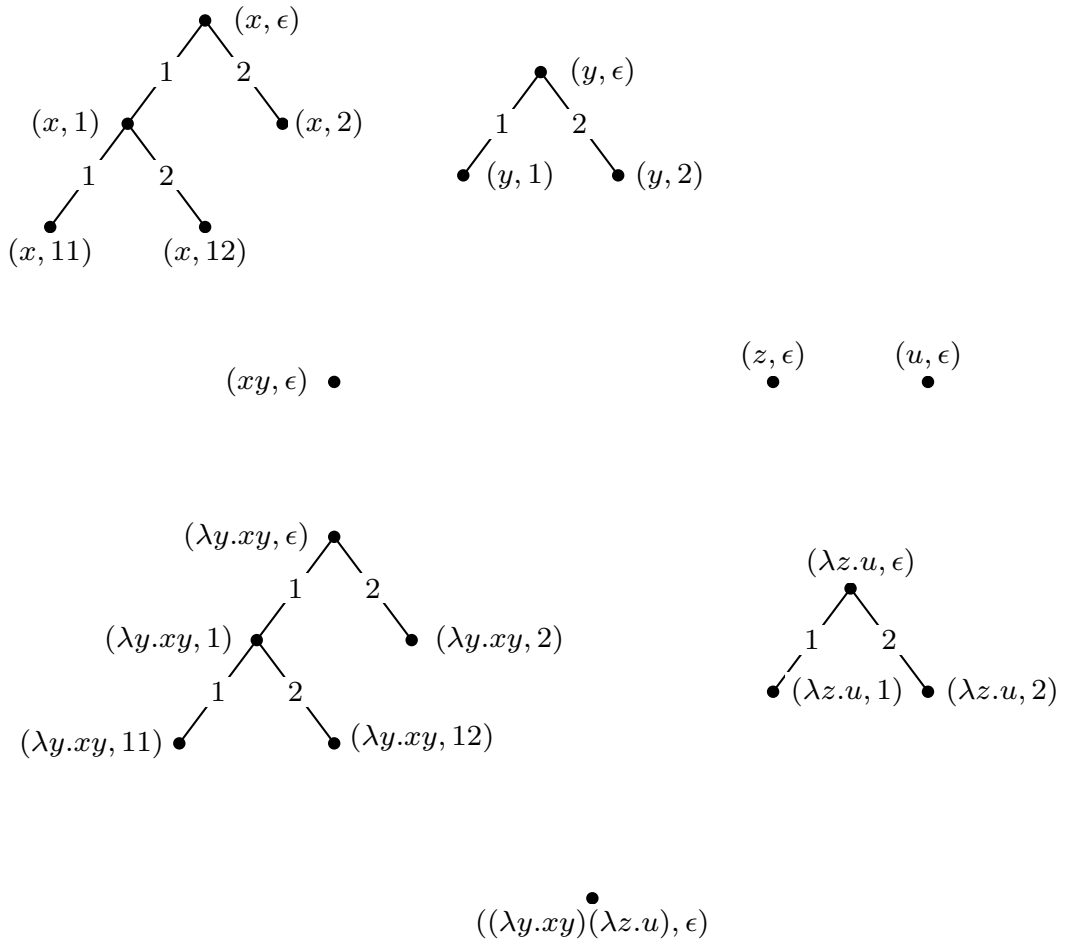
$$\dfrac{\dfrac{\dfrac{\Gamma, y : p \to q \vdash x : (p \to q) \to r \qquad \Gamma, y : p \to q \vdash y : p \to q}{\Gamma, y : p \to q \vdash xy : r}}{\Gamma \vdash \lambda y.xy : (p \to q) \to r} \qquad \dfrac{\Gamma, z : p \vdash u : q}{\Gamma \vdash \lambda z.u : p \to q}}{\Gamma \vdash (\lambda y.xy)(\lambda z.u) : r}$$

Rysunek 2.2 pokazuje prosty graf typów dla tego wyprowadzenia.

**Definicja 2.3.5.** *Pełnym grafem typów* dla wyprowadzenia $D$ nazwę graf $\mathcal{G}_D = (V, E)$ taki, że

- $V = V(\mathcal{G}^0(D))$,
- jeśli $(a \overset{l}{\longleftrightarrow} b) \in E(\mathcal{G}^0(D)))$, to $(a \overset{l}{\longleftrightarrow} b) \in E$,
- jeśli $(P, 1\rho) \in V$, $(Q, \rho) \in V$ i $PQ \in SubT(M)$, to $((P, 1\rho) \overset{e}{\longleftrightarrow} (Q, \rho))$,
- jeśli $(P, 2\rho) \in V$, $(PQ, \rho) \in V$, to $((P, 2\rho) \overset{e}{\longleftrightarrow} (PQ, \rho))$,
- jeśli $(\lambda x.P, 1\rho) \in V$, to $((\lambda x.P, 1\rho) \overset{e}{\longleftrightarrow} (x, \rho))$.

**Przykład 2.3.6.** Rysunek 2.3 pokazuje pełny graf typów wyprowadzenia z przykładu 2.3.4.




Rysunek 2.2: Prosty graf typów – przykład

Niech $F : V(\mathcal{G}_D) \to \mathbb{N}$ będzie dowolną funkcją. Określamy funkcje $F_i$ dla $i \in \mathbb{N}$:

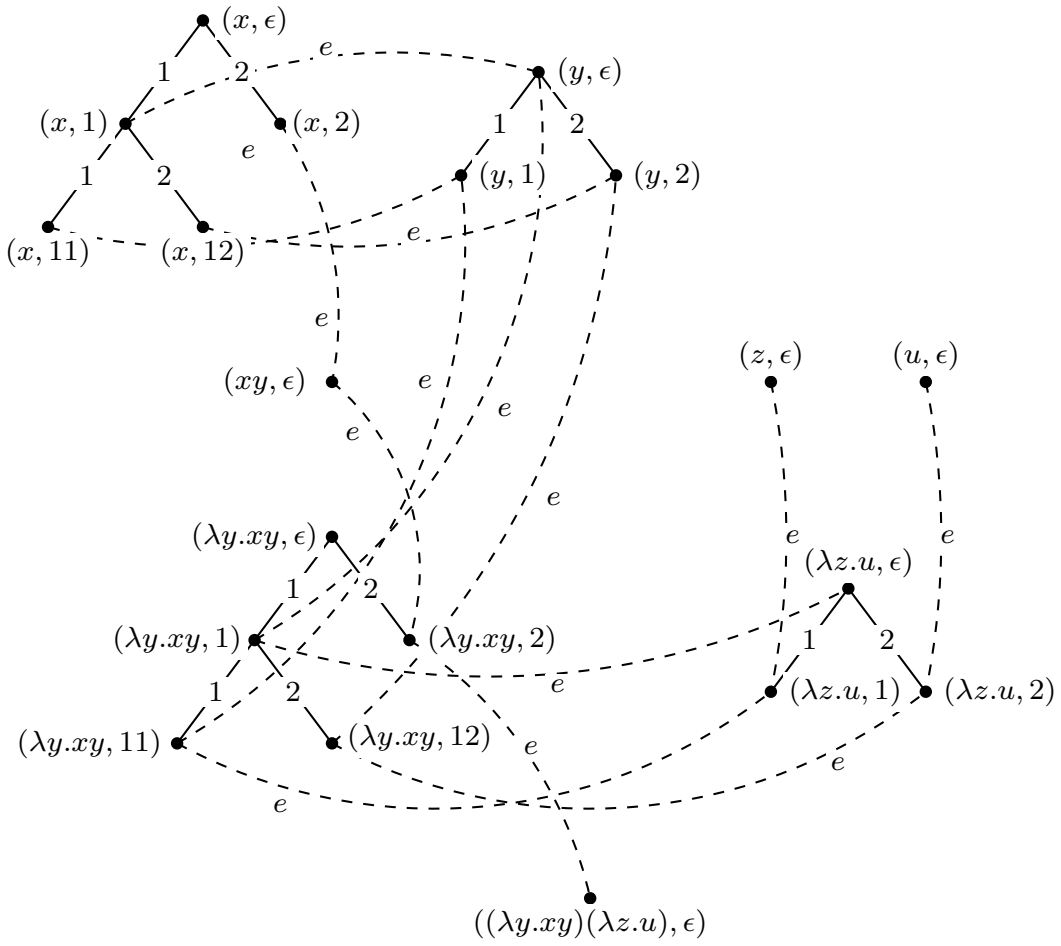
1. $F_0 = F$,
2. jeśli $a, b \in V(\mathcal{G}_D)$, $(a \overset{e}{\longleftrightarrow} b) \in E(\mathcal{G}_D)$ i $F_i(a) \neq F_i(b)$, to $F_{i+1}(a) = F_{i+1}(b) = max(F_i(a), F_i(b))$ oraz $F_{i+1}(x) = F_i(x)$ dla $x \in V(\mathcal{G}_D)$ takich, że $x \neq a$ i $x \neq b$.

Funkcję $\widehat{F}$ określamy w następujący sposób: $\widehat{F} = F_n$, gdzie $n$ jest najmniejszą liczbą taką, że $F_n = F_{n+1}$. Ponieważ $D$ jest skończone, to $\widehat{F}$ jest dobrze określone.

Niech $D$ będzie wyprowadzeniem pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$. Określam funkcję $\Theta^0 : V(\mathcal{G}_D) \to \mathbb{N}$ następująco:

- jeśli $y$ jest zmienną wolną i $(y, \rho) \in V(\mathcal{G}_D)$, to $\Theta^0((y, \rho)) = m$, gdzie $m$ jest liczbą niewiążących kwantyfikatorów stojących na pozycji $\rho$ w $\Gamma(y)$,
- jeśli $\rho \in Paths(\tau)$, to $\Theta^0((M, \rho)) = m$, gdzie $m$ jest liczbą niewiążących kwantyfikatorów stojących na pozycji $\rho$ w $\tau$,
- $\Theta^0(z) = 0$ dla pozostałych $z \in V(\mathcal{G}_D)$.

**Definicja 2.3.7.** *Funkcją ograniczenia* będę nazywać funkcję $\Theta = \widehat{\Theta^0}$.




Rysunek 2.3: Pełny graf typów – przykład

**Lemat 2.3.8.** *Funkcja ograniczenia* $\Theta$ *ma następujące własności:*

1. *Dla każdego zastosowania reguły (App) postaci* $\dfrac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma}$ *i dla każdego* $\rho \in Paths(\tau)$ *zachodzi* $\Theta((M, 1\rho)) = \Theta((N, \rho))$.
2. *Dla każdego zastosowania reguły (App) postaci* $\dfrac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma}$ *i dla każdego* $\rho \in Paths(\sigma)$ *zachodzi* $\Theta((M, 2\rho)) = \Theta((MN, \rho))$.
3. *Dla każdego zastosowania reguły (Abs) postaci* $\dfrac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma}$ *i dla każdego* $\rho \in Paths(\sigma)$ *zachodzi* $\Theta((M, \rho)) = \Theta((\lambda x.M, 2\rho))$.
4. *Dla każdego zastosowania reguły (Abs) postaci* $\dfrac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma}$ *i dla każdego* $\rho \in Paths(\tau)$ *zachodzi* $\Theta((x, \rho)) = \Theta((\lambda x.M, 1\rho))$.

Zdefiniuję pomocniczą operację $Drop_M : SubT(M) \times \{1,2\}^* \times Types \rightharpoonup Types$. Intuicyjne objaśnienie jest takie: $N$ jest podtermem $M$, $\sigma$ jest typem przypisanym $N$ w wyprowadzeniu $D$, $\rho$ jest ścieżką w $\sigma$, $\tau$ jest formułą, która występuje w $\sigma$ na pozycji $\rho$. Wtedy $Drop_M(N, \rho, \tau)$ to formuła $\tau'$, która powstaje z $\tau$ przez wytarcie jak najmniejszej liczby zewnętrznych kwantyfikatorów niewiążących w taki sposób, że na pozycji $\epsilon$ w $\tau'$ stoi nie więcej niż $\Theta(N, \rho))$ kwantyfikatorów niewiążących.

- $Drop_M(N, \rho, P(t_1, ..., t_n)) = P(t_1, ..., t_n)$,
- $Drop_M(N, \rho, \tau \to \sigma) = Drop_M(N, \rho 1, \tau) \to Drop_M(N, \rho 2, \sigma)$,
- $Drop_M(N, \rho, \forall \alpha.\tau) = Drop_M(N, \rho, \tau)$, jeśli $\alpha \notin FVar(\tau)$ i w $\tau$ na pozycji $\epsilon$ jest $\Theta((N, \rho))$ lub więcej kwantyfikatorów niewiążących,
- $Drop_M(N, \rho, \forall \alpha.\tau) = \forall \alpha.Drop_M(N, \rho, \tau)$, jeśli $\alpha \in FVar(\tau)$ lub w $\tau$ na pozycji $\epsilon$ jest mniej niż $\Theta((N, \rho))$ kwantyfikatorów niewiążących.

**Lemat 2.3.9.** *Dla dowolnego $N \in SubT(M)$ oraz $\rho$ i $\tau$ takich, że $Drop_M(N, \rho, \tau)$ jest określone, zachodzi $Drop_M(N, \rho, \tau[\alpha := t]) = Drop_M(N, \rho, \tau)[\alpha := t]$.*

*Dowód.* Indukcja ze względu na definicję operacji $Drop_M$ i budowę typu $\tau$.

- $Drop_M(N, \rho, P(t_1, ..., t_n)[\alpha := t]) = P(t_1, ...t_n)[\alpha := t] = Drop_M(N, \rho, P(t_1, ..., t_n))[\alpha := t]$.
- $Drop_M(N, \rho, (\tau \to \sigma)[\alpha := t]) = Drop_M(N, \rho, (\tau[\alpha := t]) \to (\sigma[\alpha := t])) = = Drop_M(N, \rho 1, \tau[\alpha := t]) \to Drop_M(N, \rho 2, \sigma[\alpha := t]) = Drop_M(N, \rho 1, \tau)[\alpha := t] \to Drop_M(N, \rho 2, \sigma)[\alpha := t] = Drop_M(N, \rho, \tau \to \sigma)[\alpha := t]$.
- Jeśli $\beta \notin FVar(\tau)$ i w $\tau$ na pozycji $\epsilon$ jest $\Theta((N, \rho))$ lub więcej kwantyfikatorów niewiążących, to $Drop_M(N, \rho, (\forall \beta.\tau)[\alpha := t]) = Drop_M(N, \rho, \forall \beta.(\tau[\alpha := t])) = \forall \beta.Drop_M(N, \rho, \tau[\alpha := t]) = \forall \beta.Drop_M(N, \rho, \tau)[\alpha := t] = (\forall \beta.Drop_M(N, \rho, \tau))[\alpha := t]$.
- Jeśli $\beta \in FVar(\tau)$ lub w $\tau$ na pozycji $\epsilon$ jest mniej niż $\Theta((N, \rho))$ kwantyfikatorów niewiążących, to $Drop_M(N, \rho, (\forall \beta.\tau)[\alpha := t]) = Drop_M(N, \rho, \forall \beta.\tau[\alpha := t]) = = \forall \beta.Drop_M(N, \rho, \tau)[\alpha := t] = (\forall \beta.Drop_M(N, \rho, \tau))[\alpha := t]$.

□

Niech $D$ będzie wyprowadzeniem pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$. Używając operacji $Drop_M$ zdefiniuję nowe wyprowadzenie $|D|$.

Środowisko $|\Gamma|$ jest takie, że:

- $Dom(|\Gamma|) = Dom(\Gamma)$,
- $|\Gamma|(x) = Drop_M(x, \epsilon, \Gamma(x))$ dla $x \in Dom(\Gamma)$.

Dla danego sekwentu $S$ wyprowadzenia $D$ postaci $\Gamma \vdash N : \sigma$ określam $|S|$ jako $|\Gamma| \vdash N : Drop_M(N, \epsilon, \sigma)$.

**Definicja 2.3.10.** Wyprowadzenie $|D|$ to wyprowadzenie, które powstaje z wyprowadzenia $D$ poprzez zastosowanie operacji $|\ .\ |$ do każdego sekwentu wyprowadzenia $D$ oraz usunięcie powtarzających się sekwentów.

**Lemat 2.3.11.** *Wyprowadzenie $|D|$ jest poprawnym wyprowadzeniem w typach pierwszego rzędu o takiej samej konkluzji, jak wyprowadzenie $D$.*

*Dowód.* Jest jasne, że $\mathcal{FS}_D = \mathcal{FS}_{|D|}$.

Pokażę, że zastosowanie operacji $|\ .\ |$ do kolejnych sekwentów daje albo poprawne użycie którejś z reguł przypisywania typów, albo identyczne sekwenty. Rozważam wszystkie możliwe przypadki.

- Jeśli zastosowano regułę (Ax) $\Gamma, x : \tau \vdash x : \tau$, to

$$|\Gamma, x : \tau \vdash x : \tau| = |\Gamma|, x : Drop_M(x, \epsilon, \tau) \vdash Drop_M(x, \epsilon, \tau),$$

czyli jest dobrze.

- Zastosowano regułę (Abs) postaci $\dfrac{S_1}{S_2}$, gdzie $S_1 = \Gamma, x : \sigma_1 \vdash N : \sigma_2$ i $S_2 = \Gamma \vdash \lambda x.N : \sigma_1 \to \sigma_2$. Wtedy

$$\begin{aligned} |S_1| &= |\Gamma|, x : Drop_M(x, \epsilon, \sigma_1) \vdash N : Drop_M(N, \epsilon, \sigma_2), \\ |S_2| &= |\Gamma| \vdash \lambda x.N : Drop_M(\lambda x.N, \epsilon, \sigma_1 \to \sigma_2). \end{aligned}$$

Ale $Drop_M(\lambda x.N, \epsilon, \sigma_1 \to \sigma_2) = Drop_M(\lambda x.N, 1, \sigma_1) \to Drop_M(\lambda x.N, 2, \sigma_2) = Drop_M(x, \epsilon, \sigma_1) \to Drop_M(N, \epsilon, \sigma_2)$. Ostatnia równość zachodzi na mocy lematu 2.3.8, punkty 3 i 4 oraz definicji operacji *Drop*. Zatem jest dobrze.

- Zastosowano regułę (App) postaci $\dfrac{S_1 \qquad S_2}{S}$, gdzie $S_1 = \Gamma \vdash N : \sigma_1 \to \sigma_2$, $S_2 = \Gamma \vdash L : \sigma_1$, $S = \Gamma \vdash NL : \sigma_2$. Wtedy

$$\begin{aligned} |S_1| &= |\Gamma| \vdash N : Drop_M(N, \epsilon, \sigma_1 \to \sigma_2), \\ |S_2| &= |\Gamma| \vdash L : Drop_M(L, \epsilon, \sigma_1), \\ |S| &= |\Gamma| \vdash MN : Drop_M(NL, \epsilon, \sigma_2). \end{aligned}$$

Ale $Drop_M(N, \epsilon, \sigma_1 \to \sigma_2) = Drop_M(N, 1, \sigma_1) \to Drop_M(N, 2, \sigma_2)$. Ponadto na mocy lematu 2.3.8, punkt 1 i definicji operacji $Drop_M$ zachodzi $Drop_M(N, 1, \sigma_1) = Drop_M(L, \epsilon, \sigma_1)$ oraz na mocy lematu 2.3.8, punkt 2 i definicji operacji $Drop_M$ zachodzi $Drop_M(N, 2, \sigma_2) = Drop_M(NL, \epsilon, \sigma_2)$. Zatem jest dobrze.

- Zastosowano regułę (Gen) postaci $\dfrac{S_1}{S_2}$, gdzie $S_1 = \Gamma \vdash N : \sigma$, $S_2 = \Gamma \vdash N : \forall\alpha.\sigma$. Wtedy

$$\begin{aligned} |S_1| &= |\Gamma| \vdash N : Drop_N(N, \epsilon, \sigma), \\ |S_2| &= |\Gamma| \vdash N : Drop_M(N, \epsilon, \forall\alpha.\sigma). \end{aligned}$$

Jeśli $\alpha \in FVar(\sigma)$, to $Drop_M(N, \epsilon, \forall\alpha.\sigma) = \forall\alpha.Drop_M(N, \epsilon, \sigma)$, zatem jest dobrze.

Jeśli $\alpha \notin FVar(\sigma)$ i w $\sigma$ na pozycji $\epsilon$ jest mniej niż $\Theta((N, \epsilon))$ niewiążących kwantyfikatorów, to $Drop_M(N, \epsilon, \forall\alpha.\sigma) = \forall\alpha.Drop_M(N, \epsilon, \sigma)$, czyli mamy zastosowanie reguły (Gen).

Jeśli $\alpha \notin FVar(\sigma)$ i w $\sigma$ na pozycji $\epsilon$ jest nie mniej niż $\Theta((N, \epsilon))$ niewiążących kwantyfikatorów, to $Drop_M(N, \epsilon, \forall\alpha.\sigma) = Drop_M(N, \epsilon, \sigma)$, czyli otrzymujemy identyczne sekwenty.

- Jeśli zastosowano regułę (Inst) $\dfrac{S_1}{S_2}$, gdzie $S_1 = \Gamma \vdash N : \forall\alpha.\sigma$, $S_2 = \Gamma \vdash N : \sigma[\alpha := t]$. Wtedy

$$\begin{aligned} |S_1| &= |\Gamma| \vdash N : Drop_M(N, \epsilon, \forall\alpha.\sigma), \\ |S_2| &= |\Gamma| \vdash N : Drop_M(N, \epsilon, \sigma[\alpha := t]). \end{aligned}$$

Jeżeli $\alpha \in FVar(\sigma)$, to $Drop_M(N, \epsilon, \forall\alpha.\sigma) = \forall\alpha.Drop_M(N, \epsilon, \sigma)$. Wtedy $Drop_M(N, \epsilon, \sigma[\alpha := t]) = Drop_M(N, \epsilon, \sigma)[\alpha := t]$, na mocy lematu 2.3.9, czyli dobrze.

Jeżeli $\alpha \notin FVar(\sigma)$, to $\sigma[\alpha := t] = \sigma$. Wtedy $Drop_M(N, \epsilon, \sigma[\alpha := t]) = Drop_M(N, \epsilon, \sigma)$. Jeśli $Drop_M(N, \epsilon, \forall\alpha.\sigma) = Drop(N, \epsilon, \sigma)$, to $|S_1| = |S_2|$, czyli dostajemy identyczne sekwenty. Jeśli $Drop_M(N, \epsilon, \forall\alpha.\sigma) = \forall\alpha.Drop(N, \epsilon, \sigma)$, to otrzymujemy zastosowanie reguły (Inst). Tak czy inaczej, jest dobrze.

□

Z lematu wynikają następujące wnioski.

**Wniosek 2.3.12.** *Jeżeli $\Gamma \vdash M : \tau$ jest wyprowadzalne w logice pierwszego rzędu, to istnieje takie wyprowadzenie $D$ o konkluzji $\Gamma \vdash M : \tau$, w którym dla każdego sekwentu $\Gamma \vdash N : \sigma$ i dla każdego $\rho \in Paths(\sigma)$ liczba kwantyfikatorów niewiążących występujących na pozycji $\rho$ w $\sigma$ jest nie większa niż $\Theta((N, \rho))$.*

**Wniosek 2.3.13.** *Jeżeli $\Gamma \vdash M : \tau$ jest wyprowadzalne w logice pierwszego rzędu, to istnieje takie wyprowadzenie $D$ o konkluzji $\Gamma \vdash M : \tau$, w którym dla każdego sekwentu $\Gamma \vdash N : \sigma$ i dla każdego $\rho \in Paths(\sigma)$ liczba kwantyfikatorów występujących na pozycji $\rho$ w $\sigma$ jest ograniczona przez arność podformuły występującej na pozycji $\rho$ w $\sigma$ powiększoną o liczbę $\Theta((N, \rho))$.*

## 2.4. Dowód rozstrzygalności

**Lemat 2.4.1.** *Niech $D$ będzie wyprowadzeniem pierwszego rzędu mającym własność INST-before-GEN o konkluzji $\Gamma \vdash M : \tau$ i takim, że wyprowadzenie $ST(D)$ jest proste. Załóżmy, że w sygnaturze nie ma symboli funkcyjnych o arności większej od zera. Wówczas liczba wolnych zmiennych algebraicznych występujących w wyprowadzeniu $D$ jest ograniczona i ograniczenie to zależy jedynie od konkluzji $\Gamma \vdash M : \tau$.*

*Dowód.* Dla każdej zmiennej $x \in V(M)$ możemy oszacować, ile różnych algebraicznych zmiennych wolnych występuje w $IDT_D(x)$. Jeżeli $x$ jest zmienną wolną w $M$, to $IDT_D(x) = \Gamma(x)$. Zatem znamy liczbę algebraicznych zmiennych wolnych występujących w $\Gamma(x)$. Dla zmiennej $x$ związanej w $M$ znamy $ST(IDT_D(x))$. Liczba różnych algebraicznych zmiennych wolnych występujących w $IDT_D(x)$ jest nie większa niż arność formuły $IDT_D(x)$. Dalej zauważmy, że liczba zmiennych wolnych występujących w danym środowisku $\Delta$ zależy jedynie od $\Delta(x) = IDT_D(x)$ dla $x \in Dom(\Delta)$. Możemy zsumować liczbę zmiennych wolnych występujących w $\Gamma$ oraz arności formuł $IDT_D(x)$ dla $x \in BV(M)$ i otrzymamy w ten sposób ograniczenie na liczbę zmiennych wolnych, które pojawiają się w środowiskach w wyprowadzeniu $D$.

Zauważmy, że jedyną regułą, która może wprowadzić nową algebraiczną zmienną wolną jest reguła (Inst). Ponieważ sygnatura nie zawiera symboli funkcyjnych o arności większej niż zero, to każda reguła (Inst) wprowadza co najwyżej jedną nową zmienną algebraiczną. Rozważane wyprowadzenie ma własność *INST-before-GEN*. Zatem w danym ciągu reguł (Inst) lub (Gen) liczba różnych zmiennych algebraicznych w głównym typie sekwentu najpierw rośnie (ale nie przekracza arności formuły), a następnie maleje. Liczba różnych algebraicznych zmiennych wolnych w wyprowadzeniu $D$ nie przekracza sumy liczby zmiennych wolnych występujących w $\Gamma$, arności formuł $IDT_D(x)$ dla $x \in BV(M)$ oraz arności formuł $IDT_D(N)$ dla $N \in SubT(M)$. Wszystkie te liczby są znane, bo znamy warunek brzegowy, wyprowadzenie główne $\widehat{D}_M$ oraz wiemy, że $ST(D)$ jest proste.

□

**Wniosek 2.4.2.** *Niech $D$ będzie wyprowadzeniem jak w lemacie 2.4.1. Niech $\{\alpha_1, \alpha_2, ...\}$ będzie ustalonym zbiorem zmiennych algebraicznych rozłącznym z $FVar(\Gamma)$. Wówczas istnieje wyprowadzenie $D'$ o konkluzji $\Gamma \vdash M : \tau$ i liczba $n$ taka, że $FVar(D') \subseteq FVar(\Gamma) \cup \{\alpha_1, ..., \alpha_n\}$.*

*Dowód.* Z lematu 2.4.1 wynika, że zbiór $FVar(D')$ jest skończony. Niech $n$ oznacza moc zbioru $FVar(D') \setminus FVar(\Gamma)$. Niech $S$ będzie różnowartościowym podstawieniem algebraicznym takim, że $Dom(S) = FVar(D') \setminus FVar(\Gamma)$ i $Ran(S) = \{\alpha_1, ..., \alpha_n\}$. Wówczas $S(D)$ jest szukanym wyprowadzeniem $D'$. □

**Twierdzenie 2.4.3.** *Problem sprawdzania typu dla sygnatury bez symboli funkcyjnych o arności większej od zera jest rozstrzygalny.*

*Dowód.* Niech $\Gamma \vdash M : \tau$ będzie rozważanym problemem. Przypuśćmy, że istnieje wyprowadzenie $D$ o konkluzji $\Gamma \vdash M : \tau$. Porównując $ST(\Gamma \vdash M : \tau)$ i $\mathcal{FS}_{\widehat{D}_M}$ widzimy, jakie podstawienie zastosowano do starych zmiennych typowych $\widehat{D}_M$, aby otrzymać $ST(D)$. Z lematu 2.1.11 wynika, że możemy założyć, że $ST(D)$ jest wyprowadzeniem prostym. Możemy zatem odtworzyć $ST(D)$.

Aby otrzymać poprawne wyprowadzenie w typach pierwszego rzędu, musimy w odpowiedni sposób rozmieścić reguły (Gen), (Inst) oraz kwantyfikatory i termy algebraiczne.

Z lematu 2.2.3 wiemy, że możemy ograniczyć się do wyprowadzeń mających własność *INST-before-GEN*. Zatem w każdym ciągu reguł (Inst) i (Gen) najpierw występują reguły (Inst), a następnie reguły (Gen). Jest jasne, że w poprawnym wyprowadzeniu ciąg reguł (Inst) nie może być dłuższy niż liczba kwantyfikatorów stojących na pozycji $\epsilon$ w pierwszym sekwencie tego ciągu. Z kolei ciąg reguł (Gen) nie może być dłuższy niż liczba kwantyfikatorów stojących na pozycji $\epsilon$ w ostatnim sekwencie tego ciągu. Oczywiście w ciągu reguł (Gen) i (Inst) występuje ten sam term $N$. Z wniosku 2.3.13 wiemy, że liczba kwantyfikatorów stojących na pozycji $\rho$ w typie danego termu $N$ jest ograniczona z góry przez pewną znaną liczbę, nazwijmy ją $a_{N,\rho}$. W szczególności dotyczy to pozycji $\epsilon$. Zatem każdy ciąg reguł (Gen) i (Inst) w $D$ jest nie dłuższy niż $2a_{N,\epsilon}$.

Rozważmy, ile różnych typów może zostać przypisane danemu termowi $N \in SubT(M)$. Wiemy, że liczba kwantyfikatorów stojących na pozycji $\rho$ w typie termu $N$ nie przekracza $a_{N,\rho}$. Ponadto na mocy wniosku 2.4.2 możemy założyć, że w typie tym pojawiają się tylko zmienne algebraiczne z pewnego ustalonego skończonego zbioru. Istnieje zatem tylko skończenie wiele nierównoważnych typów, które mogą zostać przypisane danemu termowi $N$.

Aby sprawdzić, czy istnieje poprawne wyprowadzenie, wystarczy zatem sprawdzić skończenie wiele wyprowadzeń.

□

## 2.5. Złożoność algorytmu

W tej części oszacuję złożoność podanego algorytmu. Algorytm polega na sprawdzeniu poprawności pewnej skończonej liczby przypadków. Najpierw zastanówmy się, ile trwa sprawdzenie jednego przypadku.

**Uwaga 2.5.1.** Niech $D$ będzie poprawnym składniowo wyprowadzeniem pierwszego rzędu rozmiaru $m$. Można sprawdzić, czy $D$ jest poprawnym wyprowadzeniem pierwszego rzędu w czasie kwadratowym względem $m$.

*Dowód.* Możemy założyć, że $D$ jest poprawnym składniowo wyprowadzeniem, bo składniową poprawność łatwo sprawdzić w czasie liniowym względem rozmiaru $D$. W naszej definicji wyprowadzenia przyjęliśmy, że każdy sekwent ma etykietę mówiącą, w wyniku której reguły powstał. Wystarczy zatem sprawdzić, czy jest to poprawne użycie podanej reguły. Łatwo zauważyć, że dla każdej reguły sprawdzenie, czy jest ona poprawnie użyta zajmuje czas kwadratowy względem rozmiaru sekwentów występujących w tej regule. Samo sprawdzenie poprawności typów dla termu zajmuje czas liniowy. Za każdym razem trzeba jeszcze sprawdzić, czy środowiska są równe, a to może zająć czas kwadratowy względem rozmiaru sekwentów. □

Zajmiemy się teraz oszacowaniem liczby przypadków, które musi zbadać algorytm.

**Uwaga 2.5.2.** Niech $\Gamma \vdash M : \tau$ będzie rozważanym warunkiem brzegowym rozmiaru $n$. Wówczas term $M$, typ $\tau$ i środowisko $\Gamma$ mają rozmiar nie większy niż $n$. W dalszych rozważaniach będziemy zakładać, że rozmiar termu $M$, typu $\tau$ i środowiska $\Gamma$ jest liniowy względem $n$. W szczególności będziemy zakładać, że każdy typ w $\Gamma$ ma rozmiar liniowy względem $n$.

**Uwaga 2.5.3.** Niech $M$ będzie lambda termem rozmiaru $n$. Wówczas liczba sekwentów w wyprowadzeniu $\widehat{D}_M$ jest liniowa względem $n$.

*Dowód.* Wyprowadzenie $\widehat{D}_M$ jest drzewem. Każdy węzeł wewnętrzny tego drzewa to pewien sekwent. Aplikacji w termie $M$ odpowiada pewien węzeł wewnętrzny stopnia dwa, lambda abstrakcji węzeł wewnętrzny stopnia jeden, a zmiennej pewien liść. Mamy zatem drzewo, w którym każdy węzeł jest stopnia co najwyżej dwa oraz jest co najwyżej $n$ liści i węzłów wewnętrznych stopnia jeden. Łatwo sprawdzić indukcyjnie, że liczba wszystkich węzłów takiego drzewa jest nie większa niż $2n$, czyli jest liniowa względem $n$. □

**Uwaga 2.5.4.** Niech $\Gamma \vdash M : \tau$ będzie sekwentem wielkości $n$. Niech $D$ będzie wyprowadzeniem pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$ o prostym $ST(D)$. Wtedy rozmiar $ST(D)$ może być wykładniczy względem $n$.

*Dowód.* Oczywiście rozmiar $ST(\Gamma \vdash M : \tau)$ jest nie większy niż $n$. Wiadomo ([5]), że aby odtworzyć $ST(D)$ trzeba wykonać unifikację pewnego zbioru równań $A$. Rozmiar tego zbioru równań jest wielomianowy względem $n$. Podstawienie unifikujące $R$ może być wykładnicze względem rozmiaru zbioru równań. W każdym sekwencie $\Delta \vdash N : \sigma$ pojawiają się typy dla zmiennych wolnych termu $N$ oraz typ termu $N$. Typy te należą do zbioru powstałego przez aplikację podstawienia unifikującego $R$ do zbioru $A$, zatem mogą być wykładniczo duże względem $n$. W dziedzinie każdego środowiska w $ST(D)$ występuje liniowo wiele zmiennych, zatem pojedynczy sekwent może mieć zatem wykładniczo duży rozmiar. Wszystkich sekwentów w $ST(D)$ jest liniowo wiele, zatem całe wyprowadzenie również może być wykładniczego rozmiaru.

□

**Lemat 2.5.5.** *Niech $\Gamma \vdash M : \tau$ będzie sekwentem rozmiaru $n$. Liczba różnych rozmieszczeń reguł (Gen) lub (Inst), które rozważa algorytm, może być rzędu $2^{P(n)}$ dla pewnego wielomianu $P$.*

*Dowód.* Wszystkich reguł (Gen) oraz (Inst) w wyprowadzeniu pierwszego rzędu $D$ jest nie więcej niż sumaryczna liczba kwantyfikatorów występujących w $D$.

Niech $N$ będzie podtermem $M$. Niech $\sigma$ będzie takim typem, który algorytm rozważa jako typ dla $N$. W typie $\sigma$ liczba kwantyfikatorów wiążących jest co najwyżej równa arności formuły $\sigma$. Na mocy lematu 2.5.4 arność formuły $\sigma$ może być wykładnicza względem $n$. Zatem

liczba kwantyfikatorów wiążących dla danego typu $\sigma$ jest może być wykładnicza względem $n$. Liczbę kwantyfikatorów niewiążących stojących na pozycji $\rho$ w $\sigma$ można oszacować z góry przez $n$. Wszystkich pozycji w $\sigma$ jest wykładniczo wiele, więc niewiążących kwantyfikatorów w $\sigma$ może być wykładniczo wiele. W typie dla ustalonego termu $N$ może być zatem eksponencjalnie wiele wszystkich kwantyfikatorów. Stąd wynika, że w wyprowadzeniu badanym przez algorytm może pojawić się wykładniczo wiele reguł (Gen) lub (Inst) z termem $N$. Mamy liniową liczbę wszystkich termów, z każdym z nich jest związana co najwyżej wykładnicza liczba reguł (Gen) lub (Inst). Razem mamy zatem rzędu $2^{P(n)}$ rozważanych rozmieszczeń (Inst) i (Gen) dla pewnego wielomianu $P$. □

Teraz chcę oszacować, ile nierównoważnych typów może mieć dany term $N$.

Najpierw oszacuję liczbę wszystkich algebraicznych zmiennych wolnych występujących w wyprowadzeniu $D$.

**Lemat 2.5.6.** *Niech $D$ będzie wyprowadzeniem pierwszego rzędu mającym własność INST-before-GEN o konkluzji $\Gamma \vdash M : \tau$ i takim, że wyprowadzenie $ST(D)$ jest proste. Niech $n$ będzie rozmiarem $\Gamma \vdash M : \tau$. Wówczas liczba wolnych zmiennych algebraicznych występujących w wyprowadzeniu $D$ jest wykładnicza względem $n$.*

*Dowód.* Jeżeli $x$ jest zmienną wolną w $M$, to $IDT_D(x) = \Gamma(x)$. Liczba algebraicznych zmiennych wolnych występujących w $IDT_D(x)$ dla wszystkich zmiennych wolnych termu $M$ jest zatem mniejsza niż $n$. Jeżeli $x$ jest zmienną związaną w $M$, to liczba algebraicznych zmiennych wolnych występujących w $IDT_D(x)$ jest nie większa niż arność $IDT_D(x)$. Wiemy, że arność $IDT_D(x)$ jest co najwyżej wykładnicza oraz zmiennych związanych w $M$ jest nie więcej niż $n$, więc algebraicznych zmiennych wolnych występujących w typach dla zmiennych jest co najwyżej wykładniczo wiele. Ponadto reguł (Inst) w wyprowadzeniu $D$ jest co najwyżej wykładniczo wiele, więc wprowadzają one co najwyżej wykładniczo wiele nowych algebraicznych zmiennych wolnych. Zatem liczba wszystkich algebraicznych zmiennych wolnych jest wykładnicza względem $n$.

□

To teraz chcę oszacować, ile nierównoważnych typów można przypisać danemu podtermowi $N$.

**Lemat 2.5.7.** *Niech $\Gamma \vdash M : \tau$ będzie sekwentem rozmiaru $n$. Niech $N$ będzie podtermem termu $M$. Liczba różnych typów, które algorytm może przypisać termowi $N$, jest rzędu $2^{2^{Q(n)}}$ dla pewnego wielomianu $Q$.*

*Dowód.* Niech $\sigma$ będzie typem, który algorytm rozważa jako przypisany $N$. Niech $m$ będzie rozmiarem $\sigma$. Wtedy arność $\sigma$ jest rzędu $m$. Typ $\sigma$ ma zatem $m$ miejsc, na których może wystąpić term algebraiczny. Tym termem może być albo zmienna wolna, albo stała, może tam również wystąpić zmienna związana. Mamy do dyspozycji wykładniczą liczbę zmiennych algebraicznych oraz liniową liczbę stałych. Zmienna związana może być związana na jednym z $m$ miejsc. Zatem różnych rozmieszczeń zmiennych i stałych jest rzędu $2^n + n + m^m$. Wiemy, że $m$ może być wykładniczo duże, zatem $2^n + n + m^m$ jest rzędu $2^{2^{R_1(n)}}$ dla pewnego wielomianu $R_1$. Ponadto trzeba uwzględnić, że niektóre kwantyfikatory mogą wiązać więcej niż jedno miejsce oraz uwzględnić typy różniące się kolejnością kwantyfikatorów. Miejsc do wiązania jest co najwyżej $m$. Te $m$ miejsc trzeba podzielić między kwantyfikatory. Podziałów zbioru $m$-elementowego jest nie więcej niż $m^m$, permutacji $m$ kwantyfikatorów jest $m!$, czyli mniej niż $2^{n^2}$.

Ponadto w typie $\sigma$ należy rozmieścić kwantyfikatory niewiążące. Kwantyfikatorów niewiążących na każdej pozycji jest nie więcej niż $n$. Mamy $m$ pozycji oraz co najwyżej $m$ kwantyfikatorów wiążących, między którymi możemy rozmieszczać kwantyfikatory niewiążące. Zatem rozmieszczeń kwantyfikatorów niewiążących jest rzędu $n^{2m} \approx 2^{2^{R_2(n)}}$ dla pewnego wielomianu $R_2$.

Podsumowując, wszystkich nierównoważnych typów dla $N$ jest rzędu $2^{2^{Q(n)}}$ dla pewnego wielomianu $Q$.

□

**Twierdzenie 2.5.8.** *Algorytm działa w czasie podwójnie wykładniczym względem rozmiaru danych wejściowych.*

*Dowód.* Niech $\Gamma \vdash M : \tau$ będzie wejściem dla algorytmu. Załóżmy, że $ST(\Gamma \vdash M : \tau)$ jest wyprowadzalne w typach prostych. Weźmy konkretne rozmieszczenie reguł (Gen) oraz (Inst), które rozważa algorytm. Policzymy, na ile różnych sposobów można wstawić w nim typy. Każdemu sekwentowi trzeba przypisać jakiś typ. Z każdym wystąpieniem termu $N$ jest związany pewien ciąg reguł (Inst) i (Gen). Dla każdego wystąpienia termu $N$ wybieram jeden typ jako początkowy typ dla tego ciągu. Dla każdego termu jest nie więcej niż podwójnie wykładniczo możliwości. Następnie trzeba wybrać kolejne typy kolejnych sekwentów tego ciągu. Jeżeli mamy regułę (Inst), to musimy wybrać podstawienie, którego dokonujemy. Podstawiamy zmienną albo stałą, więc mamy wykładniczo wiele możliwości. Wystąpień reguły (Inst) jest co najwyżej wykładniczo wiele, razem jest zatem rzędu $2^{2^{R_1(n)}}$ możliwości, dla pewnego wielomianu $R_1$. Jeżeli mamy regułę (Gen), to musimy wybrać, którą zmienną chcemy związać (możemy też zdecydować, że wprowadzamy kwantyfikator niewiążący). Wszystkich zmiennych występujących w danym typie jest co najwyżej wykładniczo wiele. Dla każdej reguły (Gen) wybieramy co najwyżej jedną z nich. Postępujemy w ten sposób dla każdej reguły (Gen). Wszystkich reguł (Gen) jest wykładniczo wiele. Razem mamy zatem rzędu $2^{2^{R_2(n)}}$ możliwości dla pewnego wielomianu $R_2$. Zatem dla każdego podtermu $N$ wybieramy jeden z podwójnie wykładniczej liczby typów jako początkowy, a następnie na $2^{2^{R_3(n)}}$ sposobów przypisujemy typy sekwentom, w których występuje term $N$. Zatem dla każdego podtermu mamy razem rzędu $2^{2^{R_4(n)}}$ . Wszystkich podtermów jest liniowo wiele, każdemu z nich przypisujemy typy w sposób niezależny. Zatem rozmieszczeń typów przy ustalonej strukturze reguł jest rzędu $2^{2^{R_5(n)}}$.

Różnych rozmieszczeń reguł (Gen) i (Inst) jest wykładniczo wiele, dla każdego z nich mamy podwójnie wykładniczą liczbę różnych rozmieszczeń typów, mamy zatem podwójnie wykładniczą liczbę różnych wyprowadzeń. Zbadanie jednego wyprowadzenia zajmuje czas co najwyżej wykładniczy, zatem sumaryczny czas działania algorytmu jest podwójnie wykładniczy. □

## 2.6. Niezależność od sygnatury

Zajmowaliśmy się problemem, w którym sygnatura była ograniczona do symboli funkcyjnych i relacyjnych występujących w warunku brzegowym. Można zapytać, czy dopuszczenie w rozwiązaniu symboli spoza tej sygnatury ma wpływ na istnienie rozwiązania. Pokażę, że wyprowadzalność nie zależy od tego czy ograniczamy sygnaturę, czy też nie.

Rozważania w podrozdziale 2.1, zwłaszcza uwaga 2.1.12 pokazują, że założenie o użyciu wyłącznie symboli relacyjnych występujących w warunku brzegowym nie wypływa na istnienie rozwiązania. W tej części pokażę, że tak jest również dla symboli funkcyjnych.

Niech $\Sigma$ będzie dowolną sygnaturą. Niech $\Sigma_0$ oznacza pewien ustalony podzbiór $\Sigma$. Niech $\gamma$ będzie ustaloną zmienną algebraiczną $\gamma \in Var$. Definiuję operację $Erase_\gamma^{\Sigma_0} : T(Var, \Sigma) \to T(Var, \Sigma_0)$, zastępującą termy spoza sygnatury $\Sigma_0$ przez zmienną $\gamma$.

- $Erase_\gamma^{\Sigma_0}(\alpha) = \alpha$, jeśli $\alpha \in Var$,
- $Erase_\gamma^{\Sigma_0}(f(t_1, ..., t_n)) = f(Erase_\gamma^{\Sigma_0}(t_1), ..., Erase_\gamma^{\Sigma_0}(t_n))$, jeśli $f \in \Sigma_0$ jest $n$-argumentowym symbolem funkcyjnym,
- $Erase_\gamma^{\Sigma_0}(g(t_1, ..., t_n)) = \gamma$, jeśli $g \notin \Sigma_0$.

**Lemat 2.6.1.** *Jeśli $s$, $t \in T(Var, \Sigma)$ i $\alpha \in Var$, to*

$$Erase_\gamma^{\Sigma_0}(s[\alpha := t]) = Erase_\gamma^{\Sigma_0}(s)[\alpha := Erase_\gamma^{\Sigma_0}(t)].$$

*Dowód.* Indukcja ze względu na budowę termu $s$.

- Jeśli $s = \alpha$, to $Erase_\gamma^{\Sigma_0}(\alpha[\alpha := t]) = Erase_\gamma^{\Sigma_0}(t) = \alpha[\alpha := \alpha := Erase_\gamma^{\Sigma_0}(t)] = Erase_\gamma^{\Sigma_0}(\alpha)[\alpha := Erase_\gamma^{\Sigma_0}(t)]$.
- Jeśli $s$ jest zmienną algebraiczną $\beta \neq \alpha$, to $Erase_\gamma^{\Sigma_0}(\beta[\alpha := t]) = Erase_\gamma^{\Sigma_0}(\beta) = \beta = \beta[\alpha := Erase_\gamma^{\Sigma_0}(t)] = Erase_\gamma^{\Sigma_0}(\beta)[\alpha := Erase_\gamma^{\Sigma_0}(t)]$.
- Jeśli $s$ jest postaci $f(t_1, ..., t_n)$, gdzie $f \in \Sigma_0$, to $Erase_\gamma^{\Sigma_0}(f(t_1, ..., t_n)[\alpha := t]) = Erase_\gamma^{\Sigma_0}(f(t_1[\alpha := t], ..., t_n[\alpha := t])) = f(Erase_\gamma^{\Sigma_0}(t_1[\alpha := t]), ..., Erase_\gamma^{\Sigma_0}(t_n[\alpha := t])) = f(Erase_\gamma^{\Sigma_0}(t_1)[\alpha := Erase_\gamma^{\Sigma_0}(t)], ..., Erase_\gamma^{\Sigma_0}(t_n)[\alpha := Erase_\gamma^{\Sigma_0}(t)]) = f(Erase_\gamma^{\Sigma_0}(t_1), ..., Erase_\gamma^{\Sigma_0}(t_n))[\alpha := Erase_\gamma^{\Sigma_0}(t)] = Erase_\gamma^{\Sigma_0}(f(t_1, ..., t_n))[\alpha := Erase_\gamma^{\Sigma_0}(t)]$.
- Jeśli $s$ jest postaci $g(t_1, ..., t_n)$, gdzie $g \notin \Sigma_0$, to $Erase_\gamma^{\Sigma_0}(g(t_1, ..., t_n)[\alpha := t]) = Erase_\gamma^{\Sigma_0}(g(t_1[\alpha := t], ..., t_n[\alpha := t])) = \gamma = \gamma[\alpha := Erase_\gamma^{\Sigma_0}(t)] = Erase_\gamma^{\Sigma_0}(g(t_1, ..., t_n))[\alpha := Erase_\gamma^{\Sigma_0}(t)]$.

□

Rozszerzamy działanie operacji $Erase$ na typy $Erase_\gamma^{\Sigma_0} : Types_{Var\setminus\{\gamma\},\Sigma} \to Types_{Var,\Sigma_0}$

- $Erase_\gamma^{\Sigma_0}(P(t_1, ..., t_n)) = P(Erase_\gamma^{\Sigma_0}(t_1), ..., Erase_\gamma^{\Sigma_0}(t_n))$,
- $Erase_\gamma^{\Sigma_0}(\tau_1 \to \tau_2) = Erase_\gamma^{\Sigma_0}(\tau_1) \to Erase_\gamma^{\Sigma_0}(\tau_2)$,
- $Erase_\gamma^{\Sigma_0}(\forall\alpha.\tau) = \forall\alpha.Erase_\gamma^{\Sigma_0}(\tau)$.

**Lemat 2.6.2.** *Niech $\sigma \in Types_{Var\setminus\{\gamma\},\Sigma}$ i niech $t \in T(Var \setminus \{\gamma\}, \Sigma)$. Wtedy*

$$Erase_\gamma^{\Sigma_0}(\sigma[\alpha := t]) = Erase_\gamma^{\Sigma_0}(\sigma)[\alpha := Erase_\gamma^{\Sigma_0}(t)].$$

*Dowód.* Indukcja ze względu na definicję operacji $Erase_\gamma^{\Sigma_0}$.

- $Erase_\gamma^{\Sigma_0}(P(t_1, ..., t_n)[\alpha := t]) = Erase_\gamma^{\Sigma_0}(P(t_1[\alpha := t], ..., t_n[\alpha := t]) = P(Erase_\gamma^{\Sigma_0}(t_1[\alpha := t]), ..., Erase_\gamma^{\Sigma_0}(t_n[\alpha := t])) = P(Erase_\gamma^{\Sigma_0}(t_1)[\alpha := Erase_\gamma^{\Sigma_0}(t)]), ..., Erase_\gamma^{\Sigma_0}(t_n)[\alpha := Erase_\gamma^{\Sigma_0}(t)]) = P(Erase_\gamma^{\Sigma_0}(t_1), ..., Erase_\gamma^{\Sigma_0}(t_n))[\alpha := Erase_\gamma^{\Sigma_0}(t)] = Erase_\gamma^{\Sigma_0}(P(t_1, ..., t_n))[\alpha := Erase_\gamma^{\Sigma_0}(t)]$,

- $Erase^{\Sigma_0}_{\gamma}((\tau_1 \to \tau_2)[\alpha := t]) = Erase^{\Sigma_0}_{\gamma}(\tau_1[\alpha := t] \to \tau_2[\alpha := t]) = Erase^{\Sigma_0}_{\gamma}(\tau_1[\alpha := t]) \to Erase^{\Sigma_0}_{\gamma}(\tau_2[\alpha := t]) = Erase^{\Sigma_0}_{\gamma}(\tau_1)[\alpha := Erase^{\Sigma_0}_{\gamma}(t)]) \to Erase^{\Sigma_0}_{\gamma}(\tau_2)[\alpha := Erase^{\Sigma_0}_{\gamma}(t)] = Erase^{\Sigma_0}_{\gamma}(\tau_1 \to \tau_2)[\alpha := Erase^{\Sigma_0}_{\gamma}(t)]$,
- $Erase^{\Sigma_0}_{\gamma}((\forall\beta.\tau)[\alpha := t]) = Erase^{\Sigma_0}_{\gamma}(\forall\beta.\tau[\alpha := t]) = \forall\beta.Erase^{\Sigma_0}_{\gamma}(\tau[\alpha := t]) = \forall\beta.Erase^{\Sigma_0}_{\gamma}(\tau)[\alpha := Erase^{\Sigma_0}_{\gamma}(t)] = (\forall\beta.Erase^{\Sigma_0}_{\gamma}(\tau))[\alpha := Erase^{\Sigma_0}_{\gamma}(t)] = (Erase^{\Sigma_0}_{\gamma}(\forall\beta.\tau))[\alpha := Erase^{\Sigma_0}_{\gamma}(t)]$.

□

Jak zwykle, $Erase^{\Sigma_0}_{\gamma}(\Gamma)$ to środowisko takie, że

- $Dom(Erase^{\Sigma_0}_{\gamma}(\Gamma)) = Dom(\Gamma)$,
- $(Erase^{\Sigma_0}_{\gamma}(\Gamma))(x) = Erase^{\Sigma_0}_{\gamma}(\Gamma(x))$ dla $x \in Dom(\Gamma)$.

Definiujemy działanie $Erase^{\Sigma_0}_{\gamma}$ na sekwentach. Dla danego sekwentu $S = \Gamma \vdash M : \tau$ definiujemy $Erase^{\Sigma_0}_{\gamma}(S) = Erase^{\Sigma_0}_{\gamma}(\Gamma) \vdash M : Erase^{\Sigma_0}_{\gamma}(\tau)$.

Niech $D$ będzie dowolnym wyprowadzeniem pierwszego rzędu o konkluzji $\Gamma \vdash M : \tau$. Niech $\gamma$ będzie zmienną algebraiczną taką, że $\gamma \notin Var(D)$. Niech $\Sigma_0$ będzie zbiorem symboli relacyjnych i funkcyjnych występujących w sekwencie $\Gamma \vdash M : \tau$.

**Definicja 2.6.3.** Niech $D$ będzie wyprowadzeniem pierwszego rzędu. Wówczas $Erase^{\Sigma_0}_{\gamma}(D)$ to wyprowadzenie, które powstaje z zastosowania operacji $Erase^{\Sigma_0}_{\gamma}$ do każdego sekwentu wyprowadzenia $D$.

**Lemat 2.6.4.** *Wyprowadzenie $Erase_{\gamma,\Sigma_0}(D)$ jest poprawnym wyprowadzeniem w typach pierwszego rzędu, używającym tylko sygnatury $\Sigma_0$ o konkluzji $\Gamma \vdash M : \tau$.*

*Dowód.* Z konstrukcji $Erase_{\gamma,\Sigma_0}(D)$ jest jasne, że $Erase^{\Sigma_0}_{\gamma}(\mathcal{FS}_D) = \mathcal{FS}_D$.

Sprawdzamy, czy użycie wszystkich reguł jest poprawne.

- Jeśli użyto reguły (Ax) $\Gamma, x : \sigma \vdash x : \sigma$, to $Erase^{\gamma}_{\Sigma_0}(\Gamma, x : \sigma \vdash x : \sigma) = Erase^{\gamma}_{\Sigma_0}(\Gamma), x : Erase^{\gamma}_{\Sigma_0}(\sigma) \vdash x : Erase^{\gamma}_{\Sigma_0}(\sigma)$, czyli jest dobrze.
- Użyto reguły (Abs) $\dfrac{S_1}{S_2}$, gdzie $S_1 = \Gamma, x : \sigma_1 \vdash N : \sigma_2$, $S_2 = \Gamma \vdash \lambda x.M : \sigma_1 \to \sigma_2$. Wtedy

$$\begin{aligned} Erase^{\gamma}_{\Sigma_0}(S_1) &= Erase^{\gamma}_{\Sigma_0}(\Gamma), x : Erase^{\gamma}_{\Sigma_0}(\sigma_1) \vdash N : Erase^{\gamma}_{\Sigma_0}(\sigma_2), \\ Erase^{\gamma}_{\Sigma_0}(S_2) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash \lambda x.M : Erase^{\gamma}_{\Sigma_0}(\sigma_1 \to \sigma_2). \end{aligned}$$

  Ale $Erase^{\gamma}_{\Sigma_0}(\sigma_1 \to \sigma_2) = Erase^{\gamma}_{\Sigma_0}(\sigma_1) \to Erase^{\gamma}_{\Sigma_0}(\sigma_2)$ na mocy definicji operacji $Erase$, więc mamy poprawne użycie reguły (Abs).
- Użyto reguły (App) $\dfrac{S_1 \qquad S_2}{S}$, gdzie $S_1 = \Gamma \vdash N : \sigma_1 \to \sigma_2$, $S_2 = \Gamma \vdash L : \sigma_1$, $S = \Gamma \vdash NL : \sigma_2$. Wtedy

$$\begin{aligned} Erase^{\gamma}_{\Sigma_0}(S_1) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash N : Erase^{\gamma}_{\Sigma_0}(\sigma_1 \to \sigma_2), \\ Erase^{\gamma}_{\Sigma_0}(S_2) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash L : Erase^{\gamma}_{\Sigma_0}(\sigma_1), \\ Erase^{\gamma}_{\Sigma_0}(S) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash NL : Erase^{\gamma}_{\Sigma_0}(\sigma_2). \end{aligned}$$

  Jak w poprzednim przypadku, $Erase^{\gamma}_{\Sigma_0}(\sigma_1 \to \sigma_2) = Erase^{\gamma}_{\Sigma_0}(\sigma_1) \to Erase^{\gamma}_{\Sigma_0}(\sigma_2)$. Jest to zatem poprawne użycie reguły (App).

- Użyto reguły (Gen) $\frac{S_1}{S_2}$, gdzie $S_1 = \Gamma \vdash N : \sigma$, $S_2 = \Gamma \vdash N : \forall\alpha.\sigma$. Wtedy

$$\begin{aligned} Erase^{\gamma}_{\Sigma_0}(S_1) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash N : Erase^{\gamma}_{\Sigma_0}(\sigma) \\ Erase^{\gamma}_{\Sigma_0}(S_2) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash N : Erase^{\gamma}_{\Sigma_0}(\forall\alpha.\sigma) \end{aligned}$$

  Zmienna $\gamma$ nie występuje w $D$, więc w szczególności $\gamma \neq \alpha$. Ponieważ $\alpha \notin FVar(\Gamma)$, więc także $\alpha \notin FVar(Erase^{\gamma}_{\Sigma_0}(\Gamma))$. Ponadto $Erase^{\gamma}_{\Sigma_0}(\forall\alpha.\sigma) = \forall\alpha.Erase^{\gamma}_{\Sigma_0}(\sigma)$, więc istotnie mamy poprawne użycie reguły (Gen).

- Użyto reguły (Inst) $\frac{S_1}{S_2}$, gdzie $S_1 = \Gamma \vdash N : \forall\alpha.\sigma$, $S_2 = \Gamma \vdash N : \sigma[\alpha := t]$. Wtedy

$$\begin{aligned} Erase^{\gamma}_{\Sigma_0}(S_1) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash N : Erase^{\gamma}_{\Sigma_0}(\forall\alpha.\sigma) \\ Erase^{\gamma}_{\Sigma_0}(S_2) &= Erase^{\gamma}_{\Sigma_0}(\Gamma) \vdash N : Erase^{\gamma}_{\Sigma_0}(\sigma[\alpha := t]) \end{aligned}$$

  Wówczas $Erase^{\gamma}_{\Sigma_0}(\forall\alpha.\sigma) = \forall\alpha.Erase^{\gamma}_{\Sigma_0}(\sigma)$ oraz $Erase^{\gamma}_{\Sigma_0}(\sigma[\alpha := t]) = Erase^{\gamma}_{\Sigma_0}(\sigma)[\alpha := Erase^{\gamma}_{\Sigma_0}(t)]$. Zatem jest to poprawne zastosowanie reguły (Inst).

□

Z lematu wynika następujący wniosek.

**Wniosek 2.6.5.** *Jeżeli $\Gamma \vdash M : \tau$ jest wyprowadzalne w logice pierwszego rzędu, to istnieje wyprowadzenie używające jedynie tych symboli funkcyjnych i relacyjnych, które występują w sekwencie $\Gamma \vdash M : \tau$.*

# Bibliografia


[1] H. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, North Holland, 1984.

[2] C. Dwork, P. C. Kanellakis, J. C. Mitchell, *On the sequential nature of unification*, Journal of Logic Programming, 1 (1984) 35–50.

[3] J. R. Hindley, *Basic Simple Type Theory*, Cambridge University Press, 1997.

[4] A. Schubert, *Type inference for first-order logic*, Proc. FOSSACS 2000, Lecture Notes in Computer Science 1784, ed. J.Tiuryn, pp. 297–314, Springer Verlag, 2000.

[5] M. Sørensen, P. Urzyczyn, *Lectures on the Curry-Howard isomorphism*, Elsevier, 2006.

[6] J. Tyszkiewicz, *Complexity of type inference*, Maszynopis, Uniwersytet Warszawski, 1988.

[7] J. B. Wells, *Typability and type checking in System F are equivalent and undecidable*, Annals of Pure and Applied Logic, 98 (1999) 111–156.