System typów F_ω

Systemy Typów 2010/11
Prowadzący: dr Dariusz Biernacki

Piotr Polesiuk
bassists@o2.pl

Małgorzata Jurkiewicz
gosia.jurkiewicz@gmail.com

Wrocław, dnia 6 lutego 2011 r.

1. Wstęp

No to na razie taki bałagan

2. System F_ω

W rozdziale tym chcielibyśmy przedstawić Państwu core F_ω :-)
Tylko ładnie i w paru zdaniach to trzeba napisać :D

2.1. Termy i typy w F_ω

System F_ω to rachunek będący rozszerzeniem λ_ω oraz systemu F. Wszystkie trzy wywodzą się z rachunku lambda z typami prostymi. Termy oraz typy definiujemy w $\lambda_\to$ następująco:

`t` ::=		*termy*
	`x`	*zmienne*
	λ`x : T.t`	*abstrakcja*
	`t t`	*aplikacja*
`T` ::=		*typy*
	`X`	*zmienna typowa*
	`T` $\to$ `T`	*typ funkcji*

2.1.1. System λ_ω

Główną cechą systemu λ_ω jest to, że oprócz termów zależnych od termów mamy typy zależne od typów, czyli możemy mówić o aplikacji i abstrakcji typowej[1]. By nam się nie pomyliło z abstrakcją na termach, zmienne typowe będziemy zaczynać dużą literą. Przykładowo `Tb` $= \lambda$`X.X` $\to$ `Bool` i λ`X.X` są abstrakcjami typowymi, ale λ`x.x` jest abstrakcją na termach. Do typu `Tb` możemy zaaplikować `Bool` i dostaniemy (λ`X.X` $\to$ `Bool`)`Bool` równoważne `Bool` $\to$ `Bool`. Jak widać użyłam słowa *równoważne*. W rachunku lambda z typami prostymi sposób konstrukcji typów gwarantował nam, że dwa typy $\texttt{T}_1$ i $\texttt{T}_2$ na pewno są różne (zakładając, że typy bazowe były sobie różne). W λ_ω jest inaczej – typy tego systemu możemy podzielić na klasy równoważności. Do klasy `Bool` $\to$ `Bool` należą również (`Tb`$^{\texttt{n}}$)`Bool` dla n naturalnego, a `T`$^{\texttt{n}}$ oznacza aplikację n typów `T`. Zauważmy, że odpowiednikiem takiej relacji równoważności na termach jest β-równoważność. W świecie typów nazwiemy taką relację $\equiv$[2]. Podobnie jak w świecie termów, typy są silnie normalizowalne i zachodzi własność Churcha-Rossera. Przez `nf(T)` oznaczamy postać normalną typu `T`. Dodatkowo wprowadzimy następującą regułę: $\frac{\Gamma\vdash \texttt{t:S} \quad \texttt{S}\equiv\texttt{T}}{\Gamma\vdash \texttt{t:T}}$ mówiącą, że jeżeli term jest typu `S`, to jest też typu dowolnego równoważnego z `S`.

Niestety, w tak zdefiniowanym systemie powstaje jeden problem. Nie chcielibyśmy, aby `Bool Bool` było dozwolone, tak samo, jak w świecie termów nie chcieliśmy, by `true true` było dozwolone. W świecie termów, by rozwiązać ten problem, wprowadziliśmy typy na termach,

[1] U Urzyczyna, w `T::K  T` to *konstruktor rodzaju*, `K` *rodzaj*, natomiast *typem* nazywamy konstruktor rodzaju $*$. My, kierowani składnią języka, wszystkie konstruktory rodzaju nazywamy typami.

[2] formalnie zdefiniujemy tą relację w rozdziale XXX

w świecie typów wprowadzimy *typy* na typach, czyli rodzaje. Piszemy, że $\mathtt{T} :: \mathtt{K}$, czyli typ $\mathtt{T}$ ma rodzaj $\mathtt{K}$. Wprowadzimy też jeden rodzaj bazowy $*$.

Wszystkie typy, jakie pojawiły się w $\lambda_{\to}$, są rodzaju $*$. Np. $\mathtt{Bool} :: *$, $\mathtt{Nat} \to \mathtt{Nat}$, $(\mathtt{Bool} \to \mathtt{Nat}) \to \mathtt{Nat} :: *$, itd. Rodzaj $* \Rightarrow *$ będzie dla funkcji z typów w typy, np. $\lambda\mathtt{X}.\mathtt{X} \to \mathtt{Bool} :: * \Rightarrow *$. $* \Rightarrow * \Rightarrow *$ bierze typ i zwraca funkcję typową, np. $\lambda\mathtt{X}.\lambda\mathtt{Y}.\mathtt{X} \to \mathtt{Y} :: * \Rightarrow * \Rightarrow *$, itd.

Teraz możemy $\lambda_{\to}$ rozszerzyć o następujące konstrukcje:

- rodzaje

$\mathtt{K} ::=$		rodzaje
	$*$	*rodzaj wszystkich typów*
	$\mathtt{K} \Rightarrow \mathtt{K}$	*rodzaj funkcji typowej*

- abstrakcję i aplikację typową na typach

$\mathtt{T} ::=$	$\cdots$	
	$\lambda\mathtt{X} :: \mathtt{K}.\mathtt{T}$	*abstrakcja typowa*
	$\mathtt{T}\ \mathtt{T}$	*aplikacji typowa*

2.1.2. System F

System F jest systemem, w którym dodatkowo, oprócz termów zależnych od termów, mamy termy zależne od typów. Wprowadzimy trzeci już rodzaj abstrakcji i aplikacji, poprzedni był w świecie typów, te będą w świecie termów. Znana jest nam funkcja identycznościowa $\lambda\mathtt{x}.\mathtt{x}$, w $\lambda_{\to}$ możemy ją napisać na wiele sposób: $\lambda\mathtt{x} : \mathtt{Bool}.\mathtt{x}$, $\lambda\mathtt{x} : \mathtt{Nat}.\mathtt{x}$, $\lambda\mathtt{x} : \mathtt{Bool} \to \mathtt{Nat}.\mathtt{x}$. W systemie F możemy wszystkie te funkcje zapisać jako: $\lambda\mathtt{X}.\lambda\mathtt{x} : \mathtt{X}.\mathtt{x}$. Zauważmy, że ten *term* przyjmuje jako pierwszy argument typ, następnie term pierwszego typu i zwraca go. Przykładem użycia takiego termu mogą być: $(\lambda\mathtt{X}.\lambda\mathtt{x} : \mathtt{X}.\mathtt{x})\ [\mathtt{Bool}]\ \mathtt{true}$, co daje $\mathtt{true}$, albo $(\lambda\mathtt{X}.\lambda\mathtt{x} : \mathtt{X}.\mathtt{x})\ [\mathtt{Nat}]\ \mathtt{1}$, co daje $\mathtt{1}$. W ten sposób powstała nam *uniwersalna* funkcja identycznościowa, której nadamy tzw. uniwersalny typ: $\lambda\mathtt{X}.\lambda\mathtt{x} : \mathtt{X}.\mathtt{x}\ :\ \forall\mathtt{X}.\mathtt{X} \to \mathtt{X}$. Dodatkowo, jako że dodaliśmy już do systemu rodzaje, napiszemy $\lambda\mathtt{X} :: *.\lambda\mathtt{x} : \mathtt{X}.\mathtt{x}\ :\ \forall\mathtt{X} :: *.\mathtt{X} \to \mathtt{X}$.

Po tym krótkim wstępie możemy już zdefiniować odziedziczone z systemu F własności takie, jak:

- abstrakcję i aplikację typową na termach

$\mathtt{t} ::=$	$\cdots$	*termy*
	$\lambda\mathtt{X} :: \mathtt{K}.\mathtt{t}$	*abstrakcja typowa*
	$\mathtt{t}[\mathtt{T}]$	*aplikacja typowa*

- typ uniwersalny

$\mathtt{T} ::=$	$\cdots$	*typy*
	$\forall\mathtt{X} :: \mathtt{K}.\mathtt{T}$	*typ uniwersalny*

2.2. Typowanie

2.2.1. Kontekst

Kontekst typowania opisany jest następującą składnią abstrakcyjną

$\Gamma ::=$		*kontekst*
	$\emptyset$	*pusty kontekst*
	$\Gamma, \mathtt{x} : \mathtt{T}$	*wiązanie typu*
	$\Gamma, \mathtt{X} :: \mathtt{K}$	*wiązanie rodzaju*

Konteksty typowania bedziemy często traktować jako skończone zbiory wiązań i będziemy używać teoriomnogościowych symboli na nich. Np. przynależność do kontekstu formalnie definiujemy jako:

$$\frac{}{B \in \Gamma, B} \qquad \frac{B \in \Gamma}{B \in \Gamma, B'}$$

Definicje pozostałych operacji teoriomnogościowych są na tyle naturalne, że zostawiamy je Czytelnikowi do uzupełnienia.

2.2.2. Podstawienia (może pójdzie gdzie indziej, jeszcze zobaczę)

Oprócz zwykłego podstawienia za zmienne, które pozostawiamy Czytelnikowi do uzupełnienia, powinniśmy zdefiniować podstawienie za zmienne typowe.

- $[\mathtt{Y} \mapsto \mathtt{T}]\mathtt{X} = \begin{cases} \mathtt{T} & Y = X \\ \mathtt{X} & \text{w.p.p} \end{cases}$
- $[\mathtt{Y} \mapsto \mathtt{T}](\mathtt{X}_1\ \mathtt{X}_2) = [\mathtt{Y} \mapsto \mathtt{T}]\mathtt{X}_1[\mathtt{Y} \mapsto \mathtt{T}]\mathtt{X}_2$
- $[\mathtt{Y} \mapsto \mathtt{T}](\mathtt{S}_1 \to \mathtt{S}_2) = [\mathtt{Y} \mapsto \mathtt{T}]\mathtt{S}_1 \to [\mathtt{Y} \mapsto \mathtt{T}]\mathtt{S}_2$
- $[\mathtt{Y} \mapsto \mathtt{T}]\forall \mathtt{X}.\mathtt{S} = \begin{cases} \forall \mathtt{X}.\mathtt{S} & Y = X \text{ lub } Y \notin FV(S) \\ [\mathtt{Y} \mapsto \mathtt{T}]\forall \mathtt{X}.\mathtt{S} & X \notin FV(\phi) \text{ i } Y \in FV(S) \end{cases}$
- $[\mathtt{Y} := \mathtt{T}]\lambda \mathtt{X}.\mathtt{S} = \begin{cases} \lambda \mathtt{X}.\mathtt{S} & Y = X \text{ lub } Y \notin FV(S) \\ [\mathtt{Y} := \mathtt{T}]\lambda \mathtt{X}.\mathtt{S} & X \notin FV(S) \text{ i } Y \in FV(S) \end{cases}$

2.2.3. Relacja $\equiv$

Jak wspomnieliśmy w rozdziale XXX, definiujemy na typach relację równoważność. $\mathtt{S}, \mathtt{S}_1, \mathtt{S}_2, \mathtt{T}, \mathtt{T}_1, \mathtt{T}_2$ to typy, $\mathtt{K}$ to rodzaj. Następujące trzy reguły:

$$\frac{}{\mathtt{T} \equiv \mathtt{T}} \qquad \frac{\mathtt{S} \equiv \mathtt{T}}{\mathtt{T} \equiv \mathtt{S}} \qquad \frac{S \equiv U \quad U \equiv T}{S \equiv T}$$

gwarantują nam równoważność relacji $\equiv$. Pozostałe reguły jak następuje:

$$\frac{\mathtt{S}_1 \equiv \mathtt{T}_1 \quad \mathtt{S}_2 \equiv \mathtt{T}_2}{\mathtt{S}_1 \to \mathtt{S}_2 \equiv \mathtt{T}_1 \to \mathtt{T}_2} \qquad \frac{\mathtt{S}_1 \equiv \mathtt{T}_1 \quad \mathtt{S}_2 \equiv \mathtt{T}_2}{\mathtt{S}_1\ \mathtt{S}_2 \equiv \mathtt{T}_1\ \mathtt{T}_2}$$

$$\frac{\mathtt{S} \equiv \mathtt{T}}{\lambda \mathtt{X} :: \mathtt{K}.\mathtt{S} \equiv \lambda \mathtt{X} :: \mathtt{K}.\mathtt{T}} \qquad (\lambda \mathtt{X} :: \mathtt{K}.\mathtt{S})\mathtt{T} \equiv [\mathtt{X} \mapsto \mathtt{T}]\mathtt{S}$$

2.2.4. Reguły rodzajowania :-)

W systemie F_ω każdemu poprawnie zbudowanemu typowi przyporządkowujemy rodzaj. Przyporządkowanie to określa relacja $(. \vdash . :: .)$ zdefiniowana następująco.

Jeżeli zachodzi $\Gamma \vdash \mathtt{T} :: \mathtt{K}$, to powiemy, że *typ* $\mathtt{T}$ *jest rodzaju* $\mathtt{K}$ *w kontekście* Γ, gdzie relacja określenia rodzaju $(. \vdash . :: .) \subseteq \Gamma \times T \times K$ jest najmniejszą relacją zamkniętą na reguły:

$$\frac{\mathtt{X} :: \mathtt{K} \in \Gamma}{\Gamma \vdash \mathtt{X} :: \mathtt{K}} \qquad \frac{\Gamma \vdash \mathtt{T}_1 :: \mathtt{K}_1 \Rightarrow \mathtt{K}_2 \quad \Gamma \vdash \mathtt{T}_2 :: \mathtt{K}_1}{\Gamma \vdash \mathtt{T}_1\mathtt{T}_2 :: \mathtt{K}_2}$$

$$\frac{\Gamma \vdash \mathtt{X} :: \mathtt{K}_1 \quad \Gamma \vdash \mathtt{T} :: \mathtt{K}_2}{\Gamma \vdash \lambda \mathtt{X} :: \mathtt{K}.\mathtt{T} :: \mathtt{K}_1 \Rightarrow \mathtt{K}_2} \qquad \frac{\Gamma \vdash \mathtt{X} :: \mathtt{K} \quad \Gamma \vdash \mathtt{T} :: *}{\Gamma \vdash \forall \mathtt{X} :: \mathtt{K}.\mathtt{T} :: *}$$

$$\frac{\Gamma \vdash \mathtt{T}_1 : * \quad \Gamma \vdash \mathtt{T}_2 : *}{\mathtt{T}_1 \to \mathtt{T}_2 : *}$$

2.2.5. Reguły typowania

Jesteśmy już gotowi przedstawić reguły typowania zdefiniowanego wyżej systemu F_ω. Każdemu poprawnie zbudowanemu termowi przyporządkowujemy typ. Przyporządkowanie to określa relacja $(. \vdash . : .)$ zdefiniowana następująco.

$$\frac{\mathtt{x} : \mathtt{T} \in \Gamma}{\Gamma \vdash \mathtt{x} : \mathtt{T}} \qquad \frac{\Gamma \vdash \mathtt{T}_1 :: * \quad \Gamma, \mathtt{x} : \mathtt{T}_1 \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \lambda \mathtt{x} : \mathtt{T}_1.\mathtt{t}_2 \ : \ \mathtt{T}_1 \to \mathtt{T}_2}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_1 \to \mathtt{T}_2 \quad \Gamma \vdash \mathtt{t}_2 : \mathtt{T}_1}{\Gamma \vdash \mathtt{t}_1 \ \mathtt{t}_2 : \mathtt{T}_2} \qquad \frac{\Gamma \vdash \mathtt{t} : \mathtt{S} \quad \mathtt{S} \equiv \mathtt{T} \quad \Gamma \vdash \mathtt{T} :: *}{\Gamma \vdash \mathtt{t} : \mathtt{T}}$$

2.3. Ewaluacja

2.3.1. Wartości

Wartości w F_ω zdefiniujemy dokładnie jak w $\lambda_\to$.

$\mathtt{v}$ ::=		*wartości*
	$\lambda \mathtt{x} : \mathtt{T}.\mathtt{t}$	*wartość abstrakcji*

2.3.2. Ewaluacja

Ewaluacja w dotychczas zdefiniowanym F_ω przebiega w sposób standardowy. W rozdziałach XXX-XXX skupimy się na rozszerzeniach minimalnej wersji F_ω i pojawią się tam nowe rzeczy. Teraz, dla czytelności, przetoczymy reguły ewaluacji dla wersji minimalnej ($\mathtt{t}_1, \mathtt{t}'_1, \mathtt{t}_2, \mathtt{t}'_2, \mathtt{t}$ to termy, $\mathtt{v}$ to wartość, $\mathtt{x}{:}\mathtt{T}$ to zmienna $\mathtt{x}$ typu $\mathtt{T}$):

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}'_1}{\mathtt{t}_1 \ \mathtt{t}_2 \longrightarrow \mathtt{t}'_1 \ \mathtt{t}_2} \qquad \frac{\mathtt{t}_2 \longrightarrow \mathtt{t}'_2}{\mathtt{v}_1 \ \mathtt{t}_2 \longrightarrow \mathtt{v}_1 \ \mathtt{t}'_2}$$

$$(\lambda \mathtt{x} : \mathtt{T}.\mathtt{t})\mathtt{v} \longrightarrow [\mathtt{x} \mapsto \mathtt{v}]\mathtt{t}$$

2.4. Inne własności F_ω

Definicja 1. Reguły przepisywania typów w systemie F_ω w wersji Curry'ego standardowe, oprócz:

$$\frac{\Gamma \vdash M : \forall X \sigma}{\Gamma \vdash M : nf(\sigma[X := \tau])}$$

Nierozstzygalne są problemy:

- sprawdzania typu: dane Γ, M, τ, Czy $\Gamma \vdash M : \tau$
- typowalność: dane M, Czy $\exists \Gamma \tau . \Gamma \vdash M : \tau$

2.5. pare słów o rozszerzeniach

3. Sładnia abstrakcyjna języka

4. Semantyka i typowanie

4.1. wyrażenia arytmetyczne i logiczne

4.2. unit i sekwencje

4.3. definicje lokalne

4.4. Rekordy

4.5. warianty

4.6. punkt stały

4.7. listy

4.8. typy egzystencjalne

4.9. typy rekurencyjne

4.10. dopasowanie wzorca

5. Rekonstrukcja typów

6. Własności i dowody

7. Praktyczne zastosowanie

8. Podsumowanie

Literatura

[1] Pierce,