Zadanie domowe 4

W dzisiejszym zadaniu domowym udowodnicie twierdzenia w logice pierwszego rzędu. Zacznijcie od przeczytania wszystkiego do Section 1.4.4 w tutorialu przed rozpoczęciem zadania. Upewnijcie się, że rozumiecie dyskutowane tam przykłady. W razie wątpliwości, skonsultujcie Coq Reference Manual celem zrozumienia nowych taktyk i komend wykorzystywanych w zadaniu.

1. Taktyki dla małych i dużych kwantyfikatorów

Poniższa tabelka podaje taktyki dla małych i dużych kwantyfikatorów w logice pierwszego rzędu:

	$\forall$(`forall`)	$\exists$(`exists`)
Reguła wprowadzania	`intro`	`exists`
Reguła eliminacji	`apply`, `apply` ... `with`$term_1 term_2 \ldots term_n$	`elim`

Oto przykładowe programy w Coq dla następujących osądów:

$$(\forall x.A \wedge B) \rightarrow (\forall x.A) \wedge (\forall x.B) true$$

$$\exists x.\neg A \rightarrow \neg \exists x.A true$$

$$\forall y.(\forall x.A) \rightarrow (\exists x.A) true$$

```
Section FirstOrder.
Variable Term : Set.
Variables A B : Term -> Prop.

Theorem forall_and :
(forall x : Term, A x /\ B x) -> (forall x : Term, A x) /\ (forall x : Term, B x).
Proof.
intro w.
split; (intro a; elim (w a); intros; assumption).
Qed.

Theorem exist_neg : (exists x : Term, ~ A x) -> (~ forall x : Term, A x).
Proof.
intro w; intro z; elim w; intros a y; elim y; apply z.
Qed.

Theorem not_weird : forall y : Term, (forall x : Term, A x) -> (exists x : Term, A x).
Proof.
intro a; intro w; exists a; apply w.
Qed.
End FirstOrder.
```

Na początek deklarujemy zbiór `Term` który będziemy używać jako zbiór termów:
`Variable Term : Set.`
Nie będziemy specyfikować elementów zbioru `Term`, jako że logika pierwszego rzędu nie zakłada żadnego szczególnego zbioru termów.

Następnie deklarujemy dwa zdania `A` oraz `B`:

`Variables A B : Term -> Prop.`
`A` oraz `B` mają zadany typ `Term -> Prop` celem zaznaczenia, że są parametryzowane przez termy, czyli elementy zbioru `Term`. Oznacza to w praktyce, że jeśli `A` zawiera jako term zmienną `x`, to piszemy `A x`, czemu nadajemy typ `Prop`, czyli zdanie. Zauważmy, że wszystkie termy, które są zmiennymi w powyższym programie w Coq mają przypisany typ `Term` tak, aby mogły być użyte jako argumenty zarówno dla `A` jak i dla `B`.

2. Zbiory, zdania i typy

Na początek trochę trudno się przyzwyczaić do różnic pomiędzy zbiorami `Set`, zdaniami `Prop` i typami `Type` w Coq, jako że wszystkie są zarówno typami jak i termami, co powoduje odrobinę zamieszania. Aby trochę ułatwić poruszanie się w systemie, odnotujmy, co następuje:

- Term dowodowy M, czyli dowód, ma określony typ A, który nazywamy zdaniem. Mamy więc relację $M : A$.
- Zdanie A ma typ `Prop`, które nazywamy **rodzajem** celem odróżnienia od typów w ogólnym znaczeniu. Tym samym mamy relację A : `Prop`, która dosłownie mówi, że A należy do zbioru zdań `Prop`.
- Term t ma pewien typ τ, który nazywamy typem danych. Mamy zatem relację $t : \tau$.
- Typ danych τ ma typ `Set`, który także będziemy nazywać **rodzajem** celem odróżnienia od typów w ogólnym sensie. Mamy więc relację τ : `Set`, która oznacza, że τ należy do zbioru typów danych `Set`.
- Zarówno `Type` jak i `Set` mają typ `Type`.

Powyższe rozważania można podsumować następującą tabelką:

term t	:	*typ danych* τ	: `Set`	: `Type`
term dowodowy M	:	*zdanie* A	: `Prop`	: `Type`

3. Deklaracje i definicje

Poniższa tabelka pokazuje w jaki sposób deklarujemy zmienne, które są termami tylko z ich typami danych oraz jak definiować zmienne, które są termami zarówno z termami, jak i z ich typami danych. Deklaracje globalne i definicje są eksportowane do zewnętrznych sekcji (zaczynających się od `Section` i kończących na `End`), w odróżnieniu od lokalnych deklaracji i definicji.

	deklaracja	*definicja*
globalna	`Parameter v` : τ, `Parameters`	`Definition c` : $\tau := t$.
lokalna	`Variable v` : τ, `Variables`	`Let c` : $\tau := t$.

Okazuje się, że powyższe definicje i deklaracje mogą być użyte nie tylko dla termów, ale także dla termów dowodowych, a nawet typów danych i zdań. Możemy, na przykład, zadeklarować typ danych następująco:
`Variable Term : Set.`
albo zdanie w taki sposób:
`Variable P : Prop.`

Dla dowodów i termów dowodowych Coq dostarcza następujących specjalnych form deklaracji i definicji. Definicja "na opak" ukrywa swój dowód M i pokazuje wyłącznie `H` oraz A do późniejszego użycia.

Definicja "przezroczysta" pokazuje również swój dowód M.

	deklaracja	*definicja*
globalna	`Axiom H : A` (`Parameter H : A` – raczej nie polecam)	`Lemma H : A. Proof M.` – "na opak" `Theorem H : A. Proof M.` – "na opak" (`Definition H : A := M.` – "przezroczysta", raczej nie polecam)
lokalna	`Hypothesis H : A, Hypotheses` (`Variable H : A` – raczej nie polecam)	`Let H : A := M.` – "przezroczysta"

4. `apply, exact` ORAZ `elim`

Dotychczas wykorzystywaliśmy te taktyki tylko dla mniennych lub etykiet. Okazuje się jednak, że ich argumentami mogą być też termy dowodowe, o ile tylko mają odpowiednie typy. Oto kilka przykładów:

- `apply (Ltn 0 (S 0)).`
  Zamiast specyfikować etykietę, używamy termu dowodowego `Ltn 0 (S 0).`
- `elim (EM (exists x, P x)).`
  Zamiast specyfikować etykietę, używamy termu dowodowego `EM (exists x, P x)` (przykład z tutoriala).
- `exact (Eqi a).`
  Zamiast specyfikować etykietę, używamy termu dowodowego `Eqi a.`

5. Własności liczb naturalnych (50% punktów).

Będziemy używać następujących aksjomatów do zdefiniowania liczb naturalnych:

$$\frac{}{Nat(\mathbf{0})true}Zero \qquad \frac{}{\forall x.Nat(x) \to Nat(\mathbf{s}(x))true}Succ$$

$$\frac{}{\forall x.Eq(x.x)true}Eq_i \qquad \frac{}{\forall x.\forall y.\forall z.(Eq(x,y) \wedge Eq(x,z) \to Eq(y,z)true}Eq_t$$

$$\frac{}{\forall x.Lt(x.\mathbf{s}(x))true}Lt_s \qquad \frac{}{\forall x.\forall y.Eq(x,y) \to \neg Lt(x,y)true}Lt_\neg$$

Powyższe aksjomaty tłumaczymy jako deklaracje w Coq w następujący sposób:

```
Variable Term : Set.

Variable O : Term.
Variable S : Term -> Term.

Variable Nat : Term -> Prop.
Variable Eq : Term -> Term -> Prop.
Variable Lt : Term -> Term -> Prop.

Hypothesis Zero : Nat O.
Hypothesis Succ : forall x : Term, Nat x -> Nat (S x).
Hypothesis Eqi : forall x : Term, Eq x x.
Hypothesis Eqt : forall (x : Term) (y : Term) (z: Term), (Eq x y /\ Eq x z) -> Eq y z.
Hypothesis Lts : forall x : Term, Lt x (S x).
Hypothesis Ltn : forall (x : Term) (y : Term), Eq x y -> ~ Lt x y.
```

Udowodnij, a następnie przetłumacz do Coq następujące twierdzenia:

$$\forall x.Nat(x) \to (\exists y.Nat(y) \wedge Eq(x,y))true$$
$$\forall x.\forall y.Eq(x,y) \to Eq(y,x)true$$
$$\neg\exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathbf{s}(\mathbf{0}))true$$

6. Dalsze własności liczb naturalnych (50% punktów).

Udowodnij następujące twierdzenia w Coq:

$$\forall x.Nat(x) \to Nat(\mathbf{s}(\mathbf{s}(x)))true$$
$$\forall x.\forall y.Lt(x,y) \to \neg Eq(x,y)true$$
$$\neg\exists x.\exists y.Eq(x,y) \wedge Lt(x,y)true$$

7. Zadanie

Ściągnijcie ze storny kursu plik:

`www.math.us.edu.pl/~pgladki/teaching/2012-2013/log_coq4.v`

i uzupełnijcie brakujące dowody z użyciem termów dowodowych. Gdy skończycie, prześlijcie uzupełniony plik na mój adres email **koniecznie** oznaczając Waszego maila tagiem `[aghlog]`. Termin oddania zadania domowego mija 7 lub 14 czerwca (w zależności, do której grupy ćwiczeniowej uczęszczacie).