Ćwiczenia

Wojciech Jedynak Paweł Wieczorek

14 października 2011

Zadanie 1. *Dodaj nowe zbiory do naszego systemu - listę parametryzowaną zbiorem A oraz drzewo binarne parametryzowane zbiorem A, tzn sformuuj wszystkie potrzebne reguły aby takie zbiory wprowadzić.*

Zadanie 2. *Na seminarium nie wprowadziliśmy zbioru odpowiadającego spójnikowi $\vee$, pokaż jak dodać taki zbiór. Zadbaj aby jego reguły odpowiadały regułom z systemu naturalnej dedukcji.*

Zadanie 3. *Ustalmy typy A oraz rodzinę $B(a)$ dla każdego $a \in A$ oraz $C(a, a')$ dla każdych $a, a' \in A$. Zbuduj termy o następujących typach:*

$$((\Pi x \in A)\ (\Pi y \in A)\ C(x,y)) \to (\Pi y \in A)\ (\Pi x \in A)\ C(x,y)$$

$$((\Sigma x \in A)\ (\Pi y \in A)\ C(x,y)) \to (\Pi y \in A)\ (\Sigma x \in A)\ C(x,y)$$

$$((\Pi x \in A)\ (\Pi y \in A)\ B(x) \to B(y)) \to (\Pi x \in A)\ (\Pi y \in A)\ \neg B(y) \to \neg B(x)$$

$$(\neg(\Sigma x \in A)\ B(x)) \to (\Pi x \in A)\ \neg B(x)$$

Zadanie 4. *Stwórz wyrażenie compose, za pomocą którego stworzymy następującą regułę:*

$$\frac{g \in A \to B \qquad f \in B \to C}{compose(f,g) \in A \to C}$$

Następnie zaproponuj regułę dla wersji z typami zależnymi i zdefiniuj odpowiednie wyrażenie composeDep.

$$\frac{g \in (\Pi x \in A) B(x) \qquad f \in (\Pi x \in A)(\Pi b \in B(x))\ C(x,b)}{composeDep(f,g) \in\ ?}$$

Zdefiniuj też pomocnicze wyrażenie apply2:

$$\frac{f \in (\Pi x \in A)(\Pi y \in B(x)) C(x,y) \qquad x \in A \qquad y \in B(x)}{apply2(f,x,y) \in\ C(x,y)}$$

Zadanie 5. *Ustalmy typ A, wyprowadź termy o następujących typach.*

$$(\Pi x \in A)\ [x =_A x]$$

$$(\Pi b \in Bool)(\Pi c \in A)\ [if\ b\ then\ c\ else\ c =_A c]$$

Zadanie 6. *Wyprowadź samodzielnie regułę dla prawa Leibniza (subst). Następnie posługując się tą regułą pokaż jak stworzyć reguły dla symetrii i przechodniości.*

$$\frac{P(x)\ set\ [x \in A] \qquad a \in A \qquad b \in A \qquad c \in [a =_A b] \qquad p \in P(a)}{subst(c, p) \in P(b)}$$

Oraz pokaż jak stworzyć cong dla poniższej reguły.

$$\frac{f \in A \to B \qquad a \in A \qquad b \in A \qquad c \in [a =_A b]}{cong(c, f) \in [apply(f, a) =_B apply(f, b)]}$$

Zadanie 7. *Zdefiniuj funkcję dodawania $add \in Nat \to Nat \to Nat$, a następnie stwórz termy o następujących typach:*

$$[apply2(add, 0, a) =_{Nat} a]$$

$$[apply2(add, a, 0) =_{Nat} a]$$

$$[apply2(add, succ(a), b) =_{Nat} succ(apply2(add, a, b))]$$

$$[apply2(add, a, b) =_{Nat} apply2(add, b, a)]$$

Powyższe równości mogą być traktowane jako specyfikacja dodawania, można teraz zauważyć że nasz system z typami zależnymi posłużył jednocześnie do zdefiniowania funkcji, zdefiniowania specyfikacji (za pomocą typów identycznościowych) oraz udowodnił że funkcja spełnia te równości – co oznacza że wszystko zostało zweryfikowane w obrębie jednego systemu.

Zadanie 8. *Niech $f \in (\Pi x \in A)B(x)$. Wyprowadź $[f =_{(\Pi x \in A)B(x)} \lambda x.apply(f, x)]$[1]. Następnie zapoznaj się z typem Eq zdefiniowanym w książce (to typ identycznościowy z silną regułą eliminacji), następnie używając tego typu wyprowadź sąd $f = \lambda x.apply(f, x) \in (\Pi x \in A)B(x)$.*

Potrzebna będzie dodatkowa reguła eliminacji dla typu funkcyjnego, jest to reguła eliminacji podobna do reguł dla typów indukcyjnych. Mamy rodzinę zbiorów C poindeksowaną parametrami konstruktora λ, tzn ciałami funkcji.

$$\frac{\begin{array}{c} f \in (\Pi x \in A)B(x) \\ C(v)\ set\ [v \in (\Pi x \in A)B(x)] \\ d(y) \in C(\lambda y)\ [y(x) \in B(x)\ [x \in A]] \end{array}}{funsplit(f, d) \in C(f)} \qquad \frac{\begin{array}{c} b(x) \in B(x)\ [x \in A] \\ C(v)\ set\ [v \in (\Pi x \in A)B(x)] \\ d(y) \in C(\lambda y)\ [y(x) \in B(x)\ [x \in A]] \end{array}}{funsplit(\lambda b, d) = d(b) \in C(\lambda b)}$$

Spróbuj zdefiniować apply za pomocą $funsplit$.

Zadanie 9. *Udowodnij następujące twierdzenia:*

$$\Big((\Pi x \in A)Eq(B(x),\ apply(f, x),\ apply(g, x))\Big) \to Eq((\Pi x \in A)B(x),\ f,\ g)$$

$$Eq((\Pi x \in A)B(x),\ f,\ g) \to \Big((\Pi x \in A)Eq(B(x),\ apply(f, x),\ apply(g, x))\Big)$$

$$[f =_{(\Pi x \in A)B(x)} g] \to \Big((\Pi x \in A)[apply(f, x) =_{B(x)} apply(g, x)]\Big)$$

Zadanie 10. *Mogąc robić wyrażenia obliczające typy zobaczyliśmy jak można definiować predykaty, przykładowo: zdefiniowaliśmy predykat $IsZero(n) = Set(natrec(n, \widehat{\top}, (x, y)\widehat{\bot}))$ który był prawdziwy tylko dla 0. Osiągnęliśmy to zwracając $\top$ dla $n = 0$ i $\bot$ dla innych wartości. Spróbuj powtórzyć dowód że 0 jest różne od $succ(n)$.*

[1] skrócony zapis: $\lambda x.e \equiv \lambda((x)e)$

Zadanie 11. *Zrób predykat Empty dla list z pierwszego zadania. Następnie, mając ustalony A, zdefiniuj wyrażenie isEmpty:*

$$isEmpty \in (\Pi xs \in List\ A).Empty(xs) \vee \neg Empty(xs)$$

Zastanów się nad różnicą pomiędzy taką funkcją a poniższą, przypomnij sobie interpretację BHK dla alternatywy.

$$isEmpty' \in List\ A \to Bool$$

Następnie, korzystając z Empty, zaprogramuj bezpieczne wersje procedur head oraz tail.

Zadanie 12. *Skonstruuj negację bitową, tzn funkcję* $negb \in Bool \to Bool$ *a następnie skonstruuj term o typie:*

$$(\Pi b \in Bool)\neg[b =_{Bool} apply(negb, b)]$$

Dodatkowo, zbuduj termy o następujących typach:

$$[apply(negb, true) =_{Bool} false]$$

$$[apply(negb, false) =_{Bool} true]$$

Zadanie 13. *Udowodnij, że nie istnieje funkcja z liczb naturalnych w ciągi zero-jedynkowe, która ma funkcję odwrotną. To jest skonstruuj term o następującym typie:*

$$\neg(\Sigma f \in N \to BinSeq)\,(\Sigma g \in BinSeq \to N)\,(\Pi s \in BinSeq)\,[s =_{BinSeq} apply(f, apply(g, s))]$$

gdzie $BinSeq$ *oznacza* $N \to Bool$.

Wskazówka: Dowód tego twierdzenia to standardowy przykład metody przekątniowej, można znaleźć rozwiązanie w Whitebooku. Trudność zadania polega na przeniesieniu rozwiązania do naszego systemu.