Obliczenia i wnioskowanie w systemie Coq
Lista 4, cd., termin: 30.04.2013

Zadanie 3 (2p.)
Napisz taktykę `simple_tauto`, która potrafi dowieść jak najwięcej formuł zbudowanych ze spójników i stałych logicznych `True, False, ->, <->, ∧, ∨, ~`. W przypadku, gdy taktyka nie potrafi udowodnić celu, nie powinna kończyć się porażką, lecz przekształcić kontekst i cel do jak najprostszej postaci. Taktyka nie może się zapętlać!
Przykład:

```
Goal forall n m p:nat,
n = m /\ m = p -> n = p.
Proof.
simple_tauto.

> 1 subgoal
  n : nat
  m : nat
  p : nat
  H : n = m
  H0 : m = p
-----------
  n = p
```

Zadanie 4 (3p.)
Napisz taktykę `collect_nats`, która zbiera w listę wszystkie zmienne typu `nat`, które są wprowadzone w kontekście.
Przykład:

```
Goal forall n m k : nat, n=m -> m=k -> n=k -> True.
Proof.
intros.
let l := collect_nats in idtac l.

> n :: m :: k :: nil
```

Zadanie 5 (4p.)
W tym zadaniu należy sformalizować semantykę naturalną prostego języka imperatywnego `While`. Dowody twierdzeń powinny być maksymalnie zautomatyzowane.

- Składnia tego języka zawiera wyrażenia arytmetyczne, wyrażenia logiczne i instrukcje.

- Wyrażenia arytmetyczne i logiczne obejmują standardowe konstrukcje. Zadeklaruj abstrakcyjny typ dla nazw zmiennych (`Parameter var : Set.`). Zdefiniuj typy `aexp` i `bexp` dla obu rodzajów wyrażeń oraz funkcje ewaluacji (`aeval : aexp -> state -> nat`, `beval : bexp -> state -> bool`, gdzie `state` jest definiowany jako `var -> int`. Zdefiniuj wygodną notację dla wyrażeń.

- Składnia instrukcji zawiera: instrukcję pustą, instrukcję przypisania, sekwencję instrukcji, instrukcję warunkową oraz pętlę "while". Zdefiniuj typ indukcyjny `command` dla instrukcji. Napisz relację `ceval : state -> command -> state -> Prop` implementującą standardową semantykę naturalną instrukcji. Zdefiniuj wygodną notację dla instrukcji oraz dla relacji `ceval`. (Zapoznaj się z konstrukcją `Inductive ... where ...` pozwalającą na jednoczesne definiowanie relacji i notacji dla niej. Możesz wtedy zapisywać term `ceval s c s'` jako np. `c | s ==> s'` od razu w definicji relacji `ceval`).

- Zdefiniuj predykat `no_loop`, który spełniają te i tylko te instrukcje, które nie zawierają konstrukcji `while`. Udowodnij, że każda instrukcja spełniająca ten predykat "zatrzymuje się".

- Udowodnij, że pętla `while true do skip` "nie zatrzymuje się".

- Udowodnij twierdzenie o niezmienniku pętli:

```
Parameter P : state -> Prop.

Theorem while_invariant :
forall (b:bexp) (c:command) (s s':state),
(forall s s':state, P s -> beval b s = true -> ceval s c s' -> P s') ->
P s -> ceval s (while b c) s' ->
P s' /\ beval b s' = false.
```