MP23 @ II UWr 29 marca 2023 r.

Lista zadań nr 6

Zadanie 1.

Przypomnij sobie definicję funkcji `map`. Następnie pokaż, że dla dowolnych funkcji f i g oraz listy xs zachodzi `(map` f `(map` g xs`))` $\equiv$ `(map (lambda (x) (`f `(`g `x)))` xs`)`. Możesz założyć, że funkcje f i g poprawnie obliczają się do wartości dla dowolnego argumentu.

Zadanie 2.

Pokaż, że funkcja `append` zawsze oblicza się do wartości, tzn. pokaż, że dla dowolnych list xs i ys istnieje lista zs taka, że `(append` xs ys`)` $\equiv$ zs.

Zadanie 3. (2 pkt)

Formuły w *negacyjnej postaci normalnej* to takie formuły rachunku zdań, w których wszystkie negacje znajdują się przy zmiennych zdaniowych. Dokładniej, formuły w negacyjnej postaci normalnej, składają się z koniunkcji, alternatywy i literałów, gdzie literały to zanegowane lub niezanegowane zmienne zdaniowe. Takie formuły można opisać następującym typem danych, sparametryzowanym typem opisującym zmienne.

```
(define-type (NNF 'v)
  (nnf-lit  [polarity : Boolean] [var : 'v])
  (nnf-conj [l : (NNF 'v)] [r : (NNF 'v)])
  (nnf-disj [l : (NNF 'v)] [r : (NNF 'v)]))
```

Flaga `polarity` w konstruktorze literału oznacza, czy zmienna jest zanegowana (wartość `#f`), czy nie (wartość `#t`). Sformułuj zasadę indukcji dla typu `NNF`.

Zadanie 4.

Zdefiniuj funkcję `neg-nnf` typu `((NNF 'a) -> (NNF 'a))` negującą formułę w negacyjnej postaci normalnej. Następnie pokaż, że `(neg-nnf (neg-nnf` φ`))` $\equiv$ φ dla dowolnej formuły φ.

Zadanie 5. (2 pkt)

Zdefiniuj funkcję `eval-nnf` typu `(('a -> Boolean) (NNF 'a) -> Boolean)` interpretującą formułę w negacyjnej postaci normalnej, przy zadanym wartościowaniu zmiennych (funkcji typu `('a -> Boolean)`). Następnie pokaż, że dla dowolnej formuły φ i wartościowania σ zachodzi `(eval-nnf` σ `(neg-nnf` φ`))` $\equiv$ `(not (eval-nnf` σ φ`))`. Możesz założyć, że funkcja σ zawsze się zatrzymuje.

Zadanie 6. (2 pkt)

Formuły rachunku zdań możemy opisać następującym typem.

```
(define-type (Formula 'v)
  (var  [var : 'v])
  (neg  [f : (Formula 'v)])
  (conj [l : (Formula 'v)] [r : (Formula 'v)])
  (disj [l : (Formula 'v)] [r : (Formula 'v)]))
```

Zdefiniuj funkcję `to-nnf` transformującą formułę do równoważnej formuły w negacyjnej postaci normalnej. Możesz definiować funkcję pomocnicze, ale wszystkie funkcje (wzajemnie) rekurencyjne powinny używać rekursji strukturalnej.

Zadanie 7. (2 pkt)

Zdefiniuj funkcję `eval-formula` interpretującą formuły z poprzedniego zadania. Następnie pokaż, że `(eval-nnf` σ `(to-nnf` φ`))` $\equiv$ `(eval-formula` σ φ`)`. Możesz założyć, że funkcja σ zawsze się zatrzymuje.

Zadanie 8.

Zdefiniuj predykat `sorted? : ((Listof Number) -> Boolean)` sprawdzający czy lista jest posortowana oraz funkcję `insert : (Number (Listof Number) -> (Listof Number))` wstawiającą element do listy posortowanej. Następnie udowodnij, że jeśli `(sorted?` xs`)` $\equiv$ `#t` to `(sorted? (insert` x xs`))` $\equiv$ `#t`.