MP24 @ II UWr 20 marca 2024 r.

Lista zadań nr 5

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 (fun 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.

```
type 'v nnf =
  | NNFLit of bool * 'v
  | NNFConj of 'v nnf * 'v nnf
  | NNFDisj of 'v nnf * 'v nnf
```

Flaga boolowska w konstruktorze literału oznacza, czy zmienna jest zanegowana (wartość **`true`**), czy nie (wartość **`false`**). Sformułuj zasadę indukcji dla typu `NNF`.

Zadanie 4.

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

Zadanie 5. (2 pkt)

Zdefiniuj funkcję `eval_nnf` typu `('a -> bool) -> 'a nnf -> bool` interpretującą formułę w negacyjnej postaci normalnej, przy zadanym wartościowaniu zmiennych (funkcji typu `'a -> bool`). 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.

```
type 'v formula =
  | Var of v
  | Neg of 'v formula
  | Conj of 'v formula * 'v formula
  | Disj of 'v formula * 'v formula
```

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 `is_sorted : int list -> bool` sprawdzający czy lista jest posortowana oraz funkcję `insert : int -> int list -> int list` wstawiającą element do listy posortowanej. Następnie udowodnij, że jeśli `is_sorted` $xs \equiv$ **`true`** to `is_sorted (insert` x xs`)` $\equiv$ **`true`**.