

## Lista zadań nr 6

### Zadanie 1.

Przypomnij sobie definicję funkcji `map`. Następnie pokaż, że dla dowolnych funkcji  $f$  i  $g$  oraz listy  $xs$  zachodzi  $(\text{map } f (\text{map } g \ xs)) \equiv (\text{map } (\lambda x. (f (g \ x))) \ xs)$ . 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  $(\text{append } xs \ ys) \equiv zs$ .

### 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 literalów, 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 literalu oznacza, czy zmienna jest zanegowana (wartość `#f`), czy nie (wartość `#t`). Sformułuj zasadę indukcji dla typu `NNF`.

### Zadanie 4.

Zdefiniuj funkcję `neg-nnf` typu  $((\text{NNF } 'a) \rightarrow (\text{NNF } 'a))$  negującą formułę w negacyjnej postaci normalnej. Następnie pokaż, że zachodzi  $(\text{neg-nnf } (\text{neg-nnf } \varphi)) \equiv \varphi$  dla dowolnej formuły  $\varphi$ .

**Zadanie 5. (2 pkt)**

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

**Zadanie 8.**

Udowodnij przytoczone na wykładzie twierdzenie o wstawianiu do listy posortowanej: jeśli  $(\text{sorted? } xs) \equiv \#t$  to  $(\text{sorted? (insert } x \text{ } xs)) \equiv \#t$ .