MP18 @ II UWr 19 marca 2018 r.

# Lista zagadnień nr 5

## Przed zajęciami

W tym tygodniu podsumowujemy pierwszą część wykładu. Będziemy pracować z bardziej skomplikowanymi strukturami danych, w szczególności *reprezentacjami wielorakimi*, formułować ich własności i dowodzić o nich twierdzenia; zastanowimy się też nad problemem alokacji pamięci, żeby móc lepiej określić jak sprawnie działają nasze programy. Przed zajęciami należy przeczytać **Rozdział 2.4** podręcznika, potrafić **sformułować i użyć twierdzenia o indukcji** dla drzewiastych struktur danych, w szczególności dla reprezentacji formuł rachunku zdań opisanych predykatem zdefiniowanym poniżej, a także rozumieć pojęcie **nieużytku** i ich wpływu na działanie i złożoność programów.

### Formuły rachunku zdań

Na zajęciach będziemy zajmować się przekształcaniem formuł rachunku zdań i różnymi ich własnościami znanymi z logiki, z poprzedniego semestru. W celu przygotowania się do zajęć należy zaznajomić się z poniższym predykatem definiującym formuły rachunku zdań, sformułować twierdzenie o indukcji z niego wynikające i rozwiązać poniższe zadania.

```
(define (var? t)
  (symbol? t))

(define (neg? t)
  (and (list? t)
       (= 2 (length t))
       (eq? 'neg (car t))))

(define (conj? t)
  (and (list? t)
       (= 3 (length t))
       (eq? 'conj (car t))))

(define (disj? t)
  (and (list? t)
       (= 3 (length t))
       (eq? 'disj (car t))))
```

```
(define (prop? f)
  (or (var? f)
      (and (neg? f)
           (prop? (neg-subf f)))
      (and (disj? f)
           (prop? (disj-left f))
           (prop? (disj-rght f)))
      (and (conj? f)
           (prop? (conj-left f))
           (prop? (conj-rght f)))))
```

### Ćwiczenie 1.

Zdefiniuj konstruktory `neg`, `conj` i `disj`, a także odpowiednie selektory odpowiadające powyższym predykatom.

### Ćwiczenie 2.

Zdefiniuj procedurę `free-vars` znajdującą zbiór *zmiennych wolnych* formuły, reprezentowany za pomocą listy. Ponieważ w formułach zdaniowych zmienne nie są wiązane, wystarczy znaleźć zbiór wszystkich zmiennych występujących w formule.

## Na zajęciach

### Ćwiczenie 3.

Poniższa procedura definiuje zbiór wszystkich wartościowań dla danego zbioru zmiennych. Zdefiniuj procedurę `eval-formula` przyjmującą formułę i wartościowanie, i obliczającą wartość logiczną formuły dla tego wartościowania. Jeśli w formule występują zmienne nie zdefiniowane w wartościowaniu, zgłoś błąd.

```
(define (gen-vals xs)
  (if (null? xs)
      (list null)
      (let*
          ((vss  (gen-vals (cdr xs)))
           (x    (car xs))
           (vst  (map (lambda (vs) (cons (list x true)  vs)) vss))
           (vsf  (map (lambda (vs) (cons (list x false) vs)) vss)))
        (append vst vsf))))
```

Następnie przy użyciu wcześniej zdefiniowanych procedur zdefiniuj procedurę `falsifiable-eval?` przyjmującą formułę logiczną, i zwracającą wartościowanie przy którym jest ona fałszywa lub `false` jeśli jest ona tautologią.

### Ćwiczenie 4.

Zdefiniuj predykat `nnf?` który zachodzi gdy jego argument jest formułą zdaniową w negacyjnej postaci normalnej. Żeby wygodnie przedstawić takie formuły najlepiej zdefiniować dodatkowy typ danych reprezentujący literały (tj. potencjalnie zanegowane zmienne zdaniowe). Rozszerz procedury `free-vars` i `eval-formula` tak żeby działały zarówno dla zwykłych formuł zdaniowych, jak i dla formuł w postaci negacyjnej.

**Ćwiczenie 5.**

Zdefiniuj procedurę `convert-to-nnf` przekształcającą formułę do równoważnej jej formuły w negacyjnej postaci normalnej. Zadbaj o to żeby translacja była *strukturalna*, tj. żeby rekurencyjnie wywoływać ją wyłącznie na podformułach argumentu. **Wskazówka:** może przydać Ci się dodatkowa procedura, która będzie *wzajemenie rekurencyjna* z procedurą `convert-to-nnf`.

**Ćwiczenie 6.**

Zaproponuj reprezentację formuł w *koniunkcyjnej postaci normalnej* i zdefiniuj odpowiedni predykat `cnf?`, a także potrzebne konstruktory i selektory. Zdefiniuj procedurę `convert-to-cnf` przekształcającą formułę w negacyjnej postaci normalnej do CNFu. Tak jak poprzednio translacja powinna być strukturalna, trudność polega na zrozumieniu co zrobić z formułami w CNFie otrzymanymi z translacji podformuł.

**Ćwiczenie 7.**

Zdefiniuj procedurę `falsifiable-cnf?` która przyjmuje formułę zdaniową, tłumaczy ją do CNFu i na tej podstawie znajduje fragment wartościowania który falsyfikuje wejściową formułę (lub zwraca `false` gdy formuła jest tautologią). Porównaj działanie dwóch procedur sprawdzających czy formuła jest tautologią.

**Ćwiczenie 8.**

Udowodnij że własność z poprzedniego twierdzenia zachodzi, tj. że dla danego wartościowania $\sigma$ wartość formuły wyliczona przez `eval-formula` jest taka sama jak po przetłumaczeniu jej do CNFu. Będziesz potrzebować dodatkowej procedury `eval-cnf` obliczającej wartość formuły w CNFie. Dowód trzeba rozbić na dwa, dowód poprawności translacji do NNFu jest łatwiejszy niż z NNFu do CNFu.

# Zadanie domowe (na pracownię)

**Uwaga:** Zadanie domowe pojawi się w późniejszym terminie (prawdopodobnie w poniedziałek wieczorem) w osobnym pliku.