Drzewa AVL

Proponujemy sformalizowanie drzew AVL oraz dowiedzenie pewnych własności tych drzew oraz operacji na nich.

Państwa zadaniem będzie:

1. zdefiniowanie typu danych reprezentującego drzewa AVL, sparametryzowanego porządkiem na elementach
2. zdefiniowanie funkcji wstawiającej element do drzewa
3. (ewentualnie) zdefiniowanie funkcji usuwającej element z drzewa
4. pokazanie, że zachowują one niezmienniki AVL
5. pokazanie, że każde drzewo AVL o wysokości h ma co najmniej F_{h+3} elementy, gdzie F_i to i-ta liczba Fibonacciego.

Bardzo ładna implementacja drzew AVL znajduje się w pliku `map.ml` w bibliotece standardowej Ocamla. Ale niestety są to drzewa, w których wysokości poddrzew różnią się co najwyżej o 2 a nie 1 jak w AVL...

Jako założenia do tego zadania proszę przyjąć następujące rzeczy:

```
Parameter X : Set.
Parameter lt : X -> X -> Prop.
Infix 1 "<" lt.

Axiom lt_total : (x,y:X){x<y}+{x=y}+{y<x}.
```

`X` to typ elementów, które będą wytępować w drzewie. `lt` to relacja $<$, czyli ostra część porządku, co Państwo muszą wyspecyfikować odpowiednimi aksjomatami. Zakładamy też, że porządek ten jest totalny i rozstrzygalny, co wyraża aksjomat `lt_total`.

Poniższy przykład pokazuje jak używać tego aksjomatu:

```
Lemma lt_dec : (x,y:X){x<y}+{~x<y}.
```

```
Intros.
Decompose Sum (lt_total x y).
...
```

Lemat `lt_dec` powinien dać się udowodnić z własności `lt`.

Ponieważ `{x<y}+{x=y}+{y<x}` to tak naprawdę `({x<y}+{x=y})+{y<x}`, użycie `Elim (lt_total x y)` byłoby niewygodne. Użyta tutaj taktyka `Decompose Sum` wykonuje eliminację typu sumowego (tutaj `sumbool`) "aż do skutku".

Poddaję również pod rozwagę, że powyższy dowód można zakończyć 4 komendami (z czego 3 to Auto) o ile ma się dobrze dobrane Hinty.

Jacek Chrząszcz, 11 kwietnia 2002 *ostatnia zmiana: 12 kwietnia 2002*