Lab5 - Zadanie domowe - Radosław Kawa

Na podstawie:

1. Alfabetu A, w którym każda litera oznacza akcję,
2. Zestawu transakcji na zmiennych
3. Słowa w oznaczające przykładowe wykonanie sekwencji akcji.

Program wyznacza:

1. Wyznacza relację zależności D.
2. Wyznacza relację niezależności I.
3. Wyznacza postać normalną Foaty FNF([w]) śladu [w]
4. Rysuje graf zależności w postaci minimalnej dla słowa w.

Relacja zależności jak i niezależności są tworzone na podstawie zdefiniowanych transakcji na zmiennych.

Postać normalna Foaty jest wyznaczana za pomocą prostego algorytmu do obliczania postaci normalnych. Algorytm ten polega na utworzeniu oznaczonych stosów dla każdego elementu alfabetu. Następnie, czytając słowo od prawej do lewej dodajemy elementy na stosy zgodnie z regułami:

- Badana litera jest dodawana na swój stos.
- Litery, które są zależne od badanej litery (inaczej wszystkie litery które w grafie zależności mają połączenie z badaną literą) na swoje stosy mają dodawaną gwiazdkę (*).

W celu odczytania postaci normalnej Foaty należy brać w pętli zbiór utworzony przez litery znajdujących się na wierzchołkach stosów. Dla każdej litery z czubka stosu, patrzymy na litery połączone w grafie zależności i dla tych liter usuwamy dodane wcześniej gwiazdki. Pętla jest powtarzana do momentu opróżnienia wszystkich stosów.

Minimalny graf zależności jest tworzony za pomocą prostego algorytmu, który idąc po słowie od prawej do lewej dodaje do zbioru MIN wszystkie wierzchołki, które odwiedził. Następnie dla każdego wierzchołka z MIN, jeżeli dany wierzchołek ze zbioru MIN jest zależny od badanego wierzchołka (badanej litery słowa) to jest on dodawany do diagramu Hassego. Po wykonaniu tego algorytmu wykonywana jest metoda transitive reduction, która usuwa wszystkie zbędne krawędzie z grafu.

Użycie programu:

Każdy program jest w formacie tekstowym:

```
A = {a, b, c, d}
w = baadcb
(a) x := x + y
(b) y := y + 2z
(c) x := 3x + z
(d) z := y − z
```

gdzie:

- A - alfabet
- w - słowo

W celu analizy danego problemu należy zaimportować moduł fnf i wywołać funkcję analyze_problem ze ścieżką pliku jako arugment pierwszy, oraz z drugim, opcjonalnym argumentem draw_graph , który korzystając z biblioteki networkx rysuje graf Hassego.

Przykład:

```
import fnf
fnf.analyze_problem("problems/problem1.txt", draw_graph=True)
```

Przykładowe wywołanie programu znajduje się w pliku main.py .

Uwaga: Przed uruchomieniem programu należy zainstalować bibliotekę networkx za pomocą polecenia pip install network

Wyniki programu:

Przykład 1:

A = {'b', 'd', 'c', 'a'}

w = baadcb

D = [('a', 'a'), ('a', 'b'), ('a', 'c'), ('b', 'a'), ('b', 'b'), ('b', 'd'), ('c', 'a'), ('c', 'c'), ('c', 'd'), ('d', 'b'), ('d', 'c'), ('d', 'd')]

I = [('a', 'd'), ('b', 'c'), ('c', 'b'), ('d', 'a')]

Word's dependence graph: {0: {1, 2, 3, 5}, 1: {2, 4, 5}, 2: {4, 5}, 3: {4, 5}, 4: set(), 5: set()}

Foata normal form: (b)(da)(a)(bc)

Hasse diagram: [(0, 1), (0, 3), (1, 2), (2, 4), (2, 5), (3, 4), (3, 5)]

Hasse diagram dot file:

```
digraph G {
    3 -> 4
    3 -> 5
```

```
    2 -> 4
    2 -> 5
    1 -> 2
    0 -> 1
    0 -> 3

    0 [label="b"]
    1 [label="a"]
    2 [label="a"]
    3 [label="d"]
    4 [label="c"]
    5 [label="b"]
}
```

Przykład 2:

A = {'c', 'd', 'f', 'b', 'e', 'a'}

w = acdcfbbe

D = [('a', 'a'), ('a', 'c'), ('a', 'f'), ('b', 'b'), ('b', 'e'), ('c', 'a'), ('c', 'c'), ('c', 'e'), ('c', 'f'), ('d', 'd'), ('d', 'f'), ('e', 'b'), ('e', 'c'), ('e', 'e'), ('f', 'a'), ('f', 'c'), ('f', 'd'), ('f', 'f')]

I = [('a', 'b'), ('a', 'd'), ('a', 'e'), ('b', 'a'), ('b', 'c'), ('b', 'd'), ('b', 'f'), ('c', 'b'), ('c', 'd'), ('d', 'a'), ('d', 'b'), ('d', 'c'), ('d', 'e'), ('e', 'a'), ('e', 'd'), ('e', 'f'), ('f', 'b'), ('f', 'e')]

Word's dependence graph: {0: {1, 3, 4}, 1: {3, 4, 7}, 2: {4}, 3: {4, 7}, 4: set(), 5: {6, 7}, 6: {7}, 7: set()}

Foata normal form: (bda)(bc)(c)(ef)

Hasse diagram: [(0, 1), (1, 3), (2, 4), (3, 4), (3, 7), (5, 6), (6, 7)]

Hasse diagram dot file:

```
digraph G {
    6 -> 7
    5 -> 6
    3 -> 4
    3 -> 7
    2 -> 4
    1 -> 3
    0 -> 1

    0 [label="a"]
    1 [label="c"]
    2 [label="d"]
    3 [label="c"]
    4 [label="f"]
    5 [label="b"]
    6 [label="b"]
```

```
    7 [label="e"]
}
```