Teoria współbieżności - Zadanie domowe 1

Opis

Program, na podstawie zadanego słowa w oraz podanych transakcji, wyznacza postać normalną Foaty (FNF) oraz podaje graf Diekerta w formacie DOT.

Program został napisany przy pomocy języka Python w wersji 3.10.12. Nie są wykorzystywane żadne zewnętrzne biblioteki, zatem nie jest wymagane specjalne środowisko wirtualne.

Wejście

Jako wejście może służyć dowolny plik tekstowy o poniższym formacie.

Dozwolony format:

Oznaczenia:

- `w` - słowo wejściowe
- `akcja i` - oznaczenie i-tej transakcji
- `modyfikowana zmienna i` - zmienna i-tej transakcji, do której następuje przypisanie
- `wyrażenie arytmetyczne i` - wyrażenie arytmetyczne i-tej transakcji.

Uwaga 1

Przed interpretacją transakcji, usuwane są wszytskie znaki białe, więc np. transakcja

```
(a) x := b + c
```

zostanie zmieniona w

```
(a)x:=b+c
```

Uwaga 2

Dozwolone są tylko jednoliterowe oznaczenia akcji oraz zmiennych.

Uwaga 3

Każda mała lub wielka litera angielskiego alfabetu w wyrażeniu arytmetycznym zostanie uznana za osobną zmienną, np. wyrażenie `xy % zv` jest zależne od zmiennych `x`, `y`, `z` oraz `v`.

Przykłady poprawnych transakcji:

- (a) x := x + y
- (b) y := yu^(u%w)
- (q) u := abcd

Dozwolony format

```
w
(akcja 1) modyfikowana zmienna 1 := wyrażenie arytmetyczne zmiennych 1
(akcja 2) modyfikowana zmienna 2 := wyrażenie arytmetyczne zmiennych 2
...
(akcja n) modyfikowana zmienna n := wyrażenie arytmetyczne zmiennych n
end
```

Przykład dozwolonego formatu:

baadcb
(a) x := x + y
(b) y := y + 2z
(c) x := 3x + z
(d) z := y – z
end

Działanie

Program po poprawnym przeczytaniu danych wejściowych wykona następujące zadania i powiadomi o ich wyniku użytkownika:

1. Wypisze interpretacje transakcji
2. Na podstawie transakcji wywnioskuje alfabet liter `A`, w którym każda litera oznacza odpowiednią transakcję
3. Wyznaczy i wypisze relacje zależności `D` i niezależności `I` między literami z `A`
4. Wyznaczy i wypisze postać normalną Foaty (FNF) słowa `w`.
    - Wykorzystany jest algorytm opisany w V. Diekert, Y. Métivier – Partial commutation and traces, Handbook of Formal Languages, Springer, 1997, strona 10
5. Wyznaczy graf Diekerta dla słowa `w`.
    - Tworzy pełny graf zależności, a następnie minimalizuje go do grafu Diekerta.
    - Minimalizacja polega na przechodzeniu po literach `x` słowa `w` od lewej do prawej i usuwaniu krawędzi, jeśli między `x` i jego sąsiadem istnieje inna ścieżka (o długości > 1)
6. Wypisze graf Diekerta w formacie DOT
    - Wyznaczony graf można wyświelić np. korzystając z GraphvizOnline

Użycie

Aby użyć programu, zaleca się uruchomienie go bezośrednio przy użyciu terminala. Dane wejściowe można przekierować z pliku tekstowego na standardowe wejście programu. Przygotowane zostały pliki z przykładowymi wejściami.

Przykład użycia

Wywołanie w terminalu z przekierowaniem pliku na standardowe wejście programu:

```
cat test1.txt | python3 main.py
```

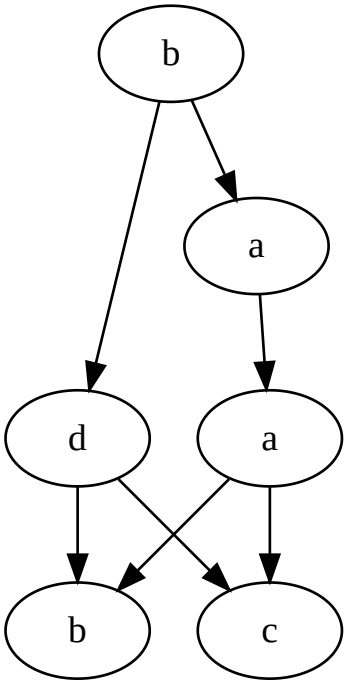
Wyniki dla przykładowych danych

test1.txt

Dla wywołania `cat test1.txt | python3 main.py` otrzymujemy:

```
Read word: 'baadcb'

Parsed actions:
action: a       modified variable: x    used variables: x,y
action: b       modified variable: y    used variables: y,z
action: c       modified variable: x    used variables: x,z
action: d       modified variable: z    used variables: y,z

Alphabet A (actions): ['a', 'b', 'c', 'd']

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')]

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

Diekert graph:
digraph Diekert {
        1 -> 2
        1 -> 4
        2 -> 3
        3 -> 6
        3 -> 5
        4 -> 6
        4 -> 5
        1[ label=b ]
        2[ label=a ]
        3[ label=a ]
        4[ label=d ]
        5[ label=c ]
        6[ label=b ]
}
```

Wizualizacja grafu Diekerta:



test2.txt

Dla wywołania `cat test2.txt | python3 main.py` otrzymujemy:

```
Read word: 'acdcfbbe'

Parsed actions:
action: a       modified variable: x    used variables: x
action: b       modified variable: y    used variables: y,z
action: c       modified variable: x    used variables: x,z
action: d       modified variable: w    used variables: w,v
action: e       modified variable: z    used variables: y,z
action: f       modified variable: v    used variables: x,v

Alphabet A (actions): ['a', 'b', 'c', 'd', 'e', 'f']

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')]

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

Diekert graph:
digraph Diekert {
        1 -> 2
        2 -> 4
```

```
        3 -> 5
        4 -> 5
        4 -> 8
        6 -> 7
        7 -> 8
        1[ label=a ]
        2[ label=c ]
        3[ label=d ]
        4[ label=c ]
        5[ label=f ]
        6[ label=b ]
        7[ label=b ]
        8[ label=e ]
}
```

Wizualizacja grafu Diekerta: