Laboratorium 5

Mateusz Cyganek

Realizacja zadania

Program zrealizowano w .NET 6.0 i Python 3.10.

Część .NET dla podanego pliku z danymi wejściowymi tworzy plik z rezultatami w którym znajduje się:

- Alfabet A
- Słowo w
- Produkcje
- Relacja zależności D
- Relacja niezależności I
- Postać normalna Foaty

Część Python generuje obraz grafu zależności.

Projekt .NET

Projekt .NET zawiera 6 plików .cs. Większość klas i metod zawiera opisy i komentarze.

Program

Klasa ta steruje działaniem całego programu, obsługuje wyjątki i wywołuje potrzebne metody.

```
// stwórz alfabet słowo i preodukcje na podstawie pliku
var serializer = new DataSerializer(inFile);
// stwórz macierz zależności na podstawie produkcji
var dependencies = new DependencyMatrix(serializer.Productions);
// stwórz FNF na podstawie słowa i macierzy zależności
var fnf = new FoatsNormalForm(serializer.Word, dependencies);
```

DataSerialiser

Klasa ta serializuje dane z pliku wejścia na użyteczne typy i klasy języka C#.

```
// lines to enumerator kolejnych linijek pliku wejściowego

while (lines.MoveNext())
    if (TrySetAlphabet(lines.Current))
        break;

while (lines.MoveNext())
    if (TrySetWord(lines.Current))
        break;

while (lines.MoveNext())
    AddProduction(lines.Current);

Validate();
```

DependencyMatrix

Klasa ta buduje macierz zależności na podstawie produkcji.

```
_alphabet = productions.Keys.ToArray();

for (var a = 0; a < _alphabet.Length; a++)
for (var b = a; b < _alphabet.Length; b++)
{
    char k1 = _alphabet[a], k2 = _alphabet[b];

    if (productions[k1].CheckDependency(productions[k2]))
        AddDependency(k1, k2);
}
```

FoatsNormalForm

Klasa ta buduje FNF na podstawie podanego słowa i macierzy zależności.

```
var chars = word.Select(x => new WordElement(x, true)).ToArray();

for (var i = 0; i < word.Length; i++)
{
    if (chars[i].Used) continue;                    // jeżeli już uśyliśmy to pomijamy

    Mask(i, chars, dependencies);                   // sprawdzamy które elementy są zależne

    var newLayer = new List<char>() {chars[i].Name}; // dodajemy obecne słowo
    chars[i].Used = true;                           // i ustawiamy je na użyte

    for (var j = i + 1; j < word.Length; j++)       // dla wszystkich kolejnych elementów słowa
        if (chars[j].Mask && !chars[j].Used)        // sprawdzamy czy są niezamaskowane i nie użyte
        {
            Mask(j, chars, dependencies);           // wykonujemy maskowanie dla elementów które
                                                    // mogą być zależne od obecnego
            newLayer.Add(chars[j].Name);            // dodajemy element
            chars[j].Used = true;                   // i ustawiamy na użyte
        }

    _fnf.Add(newLayer);
    ResetMask(chars);
}


private void Mask(int startIndex, WordElement[] word, DependencyMatrix dependencies)
{
    word[startIndex].Mask = false;

    for (var i = startIndex + 1; i < word.Length; i++)
        if (!word[i].Mask || dependencies.IsPairDepended(word[startIndex].Name, word[i].Name))
            Mask(i, word, dependencies);
}
```

Python

Skrypt pythona'a, wywoływany przez program z ścieżką do pliku z wynikiem działania generuje obrazek z grafem zależności. Plik zawiera komentarze.

Zależności skryptu:

- Os
- Sys
- Matplotlib.pyplot
- Networkx

Funkcja Generująca graf przekształca wejście postaci '(ab)(cd)(ef)' na listę list gdzie pierwszy indeks oznacza warstwę, a zawartość listy pod danym indeksem, wierzchołki niezależne które zostaną pokolorowane jednym kolorem.

```
def MakeGraph(line):
    line = line.strip()
    line = line.replace("(", "")
    g, i, chDict = [["S!"], []], 1, dict()

    for ch in line:
        if ch == ")":
            g.append([])
            i += 1
        else:
            v = 1 if not ch in chDict else chDict[ch] + 1
            chDict.update({ch: v})
            g[i].append(f"{ch}{chDict[ch]}")
    g.pop()
    return g
```

Wywołanie programu

Program należy wywołać uruchamiając w terminalu program z ścieżką pliku zwierającego dane wejściowe.

```
C:\lab5> .\Lab5_NET_Relase\Lab5_NET.exe "C:lab5\input.txt"
```

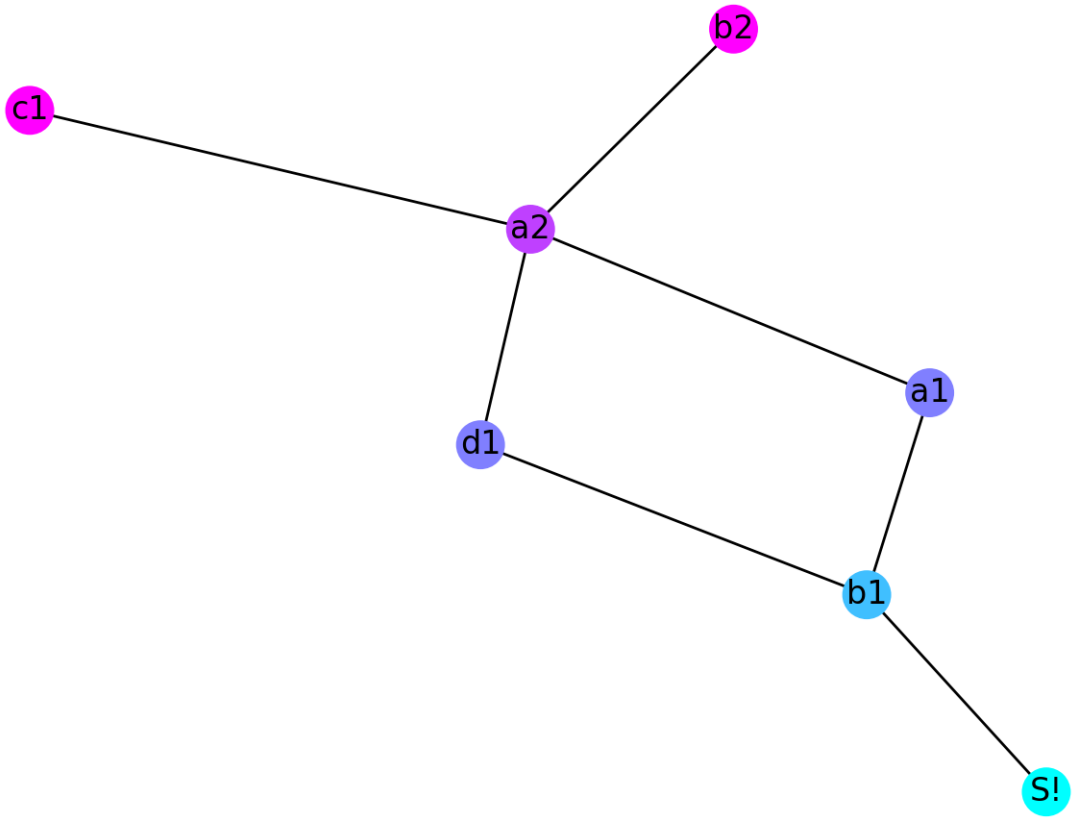
Rezultaty

Jeżeli dane podane w pliku były poprawne program zwróci następujące informacje:

Zestaw 1

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

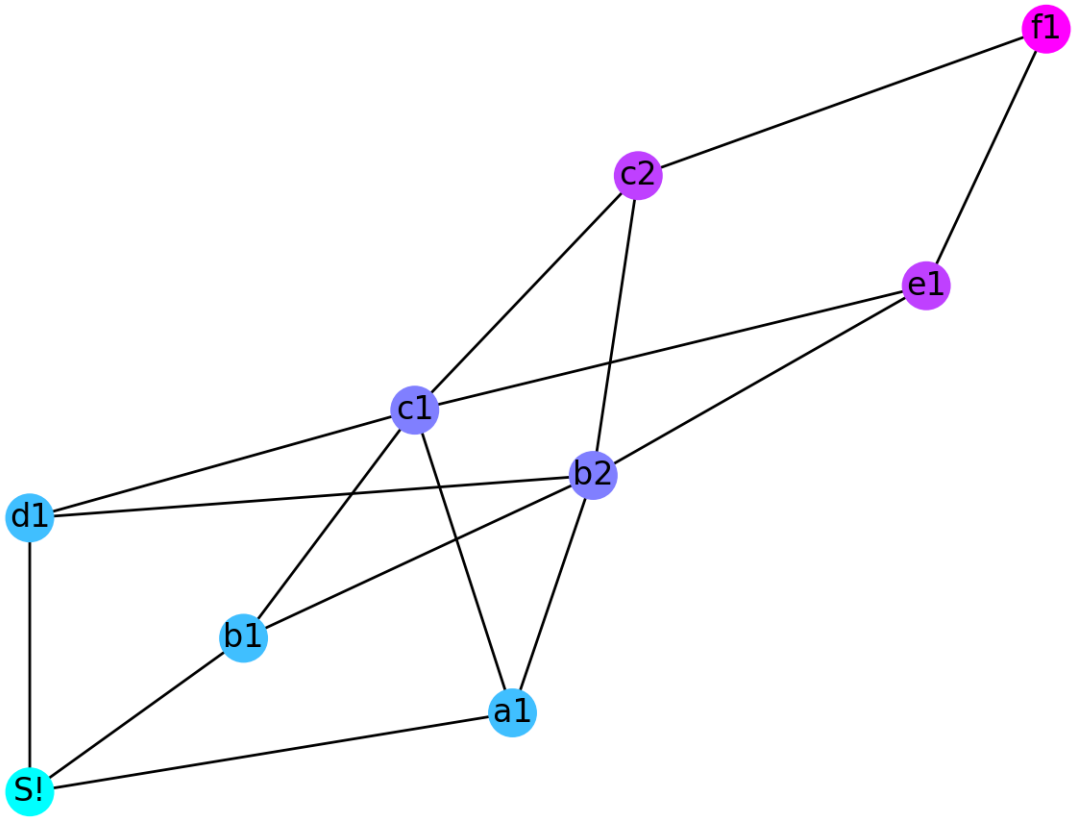
```
A = {a, b, c, d}
w = baadcb
a) x = x@y
b) y = y@z
c) x = x@z
d) z = y@z
D = sym{(a, c), (b, d)}
I = sym{(a, b), (a, d), (b, c), (c, d)}
FNF([w]) = (ba)(ad)(cb)
Wynik zapisano do: 'C:\lab5\input_results.txt'.
Graf zapisano do: 'C:\lab5\input_results_img.png'.
```


b2
c1
a2
a1
d1
b1
S!

Zestaw 2

```
A = { a, b, c, d, e, f }
w = acdcfbbe
a) x = x + 1
b) y = y + 2z
c) x = 3x + z
d) w = w + v
e) z = y - z
f) v = x + v
```

```
A = {a, b, c, d, e, f}
w = acdcfbbe
a) x = x
b) y = y@z
c) x = x@z
d) w = w@v
e) z = y@z
f) v = x@v
D = sym{(a, c), (a, f), (b, e), (c, f)}
I = sym{(a, b), (a, d), (a, e), (b, c), (b, d), (b, f), (c, d), (c, e), (d,
e), (d, f), (e, f)}
FNF([w]) = (adb)(cb)(ce)(f)
Wynik zapisano do: 'C:\lab5\input_results.txt'.
Graf zapisano do: 'C:\lab5\input_results_img.png'.
```


f1
c2
e1
c1
b2
d1
b1
a1
S!