VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATINĖS INFORMATIKOS KATEDRA

Robertas Stankevič
Bioinformatikos studijų programos
IV kurso studentas

Dirbtinis intelektas.
Laboratorinis darbas Nr. 2.
Tiesioginio išvedimo programa Perl kalba

Vadovas Vytautas Čyras

Vilnius 2015

Turinys

1. Įvadas

Tiesioginis išvedimas (angl. *Forward chaining*) yra vienas iš dviejų pagrindinių dirbtinio intelekto samprotavimo būdų produkcijų sistemoje.
Produkcijų sistema yra sudaryta iš N produkcijų, kurių kiekviena sudaryta iš priežastinių faktų aibės, ir išeigos fakto – konsekvento, kuris gaunamas panaudojus šią produkciją. Tokia produkcija (taisyklė) vaizduojama taip:

Ri : A1, A2 … An → B

Čia “Ri” – produkcijos indeksas, seka “A” – priežastinių faktų aibė, “B” - konsekventas.

Dirbtinio intelekto samprotavimo uždavinys yra surasti nepasikartojančią seką produkcijų, kurios faktų sistemą iš pradinės būsenos pervestų į terminalinę būseną, arba išsiaiškintų, kad terminalinės būsena nepasiekiama.
Tiesioginio išvedimo užduotis yra surasti tikslą, prieš tai žinant faktus.
Vykdomos iteracijos, ir kiekvienoje paeiliui ieškoma produkcijos iš produkcijų sąrašo, kurią galima būtų pritaikyti, t.y. kurioje būtų visi reikalingi faktai (prerekvizitai), be to išeigos fakto (išvados) neturi būti žinoma.
Pritaikant produkciją, naujas faktas (išvada) pridedamas prie esamų faktų aibės. Antrąsyk ta pati produkcija nebetaikoma.
Laboratorinio darbo tikslas yra parašyti tiesioginio išvedimo programą, aprašyti jos dalis komentarais, paaiškinti veikimo principą, o programa turi atvaizduoti išsamų jos vykdomų veiksmų protokolą.
Mano programa rašyta Perl programavimo kalba.
Žemiau sekančiuose skyriuose pateikiu algoritmo pseudokodą, Perl programos tekstą, pavyzdinių įvesčių ir

išvesčių, bei kelias schemas dvidalio grafo pavidalu.

Laboratorinis darbas atliktas pasiremiant A. Merkio, 2010-ųjų metų laboratoriniu darbu “Tiesioginio ir atbulinio išvedimo programos Perl kalba”.

Grafikai braižyti naudojantis programa “yEd”.

2. Algoritmo pseudokodas pagal R. E. Neapolitan straipsnį

Algoritmo įvestis:

1. *rules* – produkcijų sąrašas;
2. *assertion_list* – pradinių faktų sąrašas;
3. *goal* – tikslas.

Algoritmo išvestis:

1. *assertion_list* – visų produkcijų sistema išvedamų faktų sąrašas.

Algoritmo pseudokodas:

```
Forward-chaining(rules, assertion_list, goal)
 1. R := first of rules
 2. while more rules and goal not in assertion_list
 3.       begin
 4.            if R's premises are in assertion_list
 5.            then begin
 6.                 if R's condition is not in assertion_list
 7.                 then begin
 8.                      add R's conclusion to assertion_list
 9.                      R := first rule;
10.                 end
11.                 else R := next rule
12.            end
13.            else R := next rule
14.       end
```

3. Pavyzdžiai

3.1. Pradinis faktas konsekvente.

Pavyzdys su septyniomis produkcijomis.

Failo turinys:

```
Pavyzdys 3.
1) Taisyklės:
A L       // R1: A -> L
L K       // R2: L -> K
D A       // R3: D -> A
D M       // R4: D -> M
F B Z     // R5: F, B -> Z
C D F     // R6: C, D -> F
A D       // R7: A -> D

2) Faktai:
A B C

3) Tikslas:
Z
```

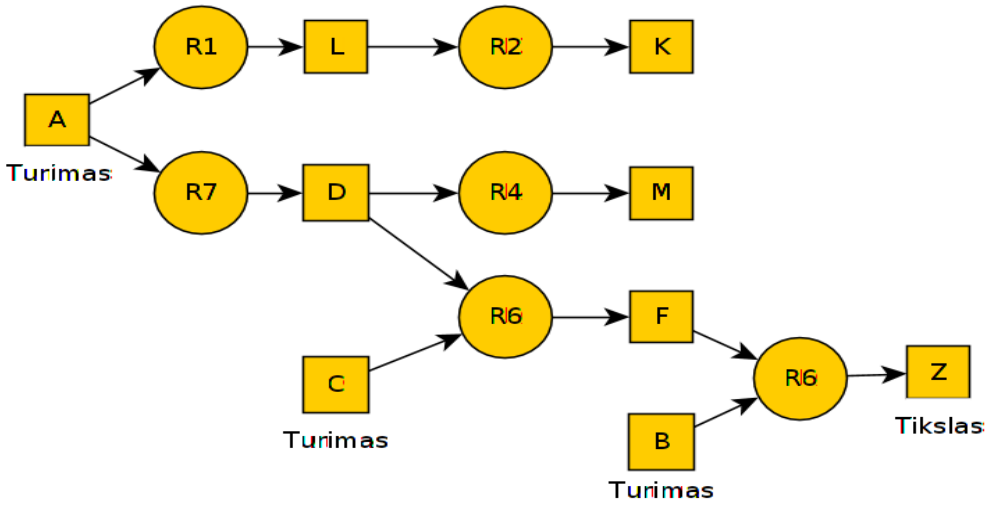
Programos rezultatas:

```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

Įvestos taisyklės:
    R1 : A -> L
    R2 : L -> K
    R3 : D -> A
    R4 : D -> M
    R5 : F, B -> Z
    R6 : C, D -> F
    R7 : A -> D
Pradiniai faktai: {A, B, C}
Tikslas: Z
--------------------------------------------------
2) Sprendimas:

Iteracija nr. 1:
    1. Taisyklė "R1 : A -> L" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, L}
--------------------
Iteracija nr. 2:
    1. Taisyklė "R1 : A -> L" netaikoma, nes pažymėta 'flag1'.
    2. Taisyklė "R2 : L -> K" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, L, K}
--------------------
```

tęsinys ...

```
Iteracija nr. 3:
    1. Taisyklė "R1 : A -> L" netaikoma, nes pažymėta 'flag1'.
    2. Taisyklė "R2 : L -> K" netaikoma, nes pažymėta 'flag1'.
    3. Taisyklė "R3 : D -> A" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    4. Taisyklė "R4 : D -> M" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    5. Taisyklė "R5 : F, B -> Z" netaikoma, nes nėra 'prerekvizito' (F) sąraše.
    6. Taisyklė "R6 : C, D -> F" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    7. Taisyklė "R7 : A -> D" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, L, K, D}
--------------------
Iteracija nr. 4:
    1. Taisyklė "R1 : A -> L" netaikoma, nes pažymėta 'flag1'.
    2. Taisyklė "R2 : L -> K" netaikoma, nes pažymėta 'flag1'.
    3. Taisyklė "R3 : D -> A" netaikoma, nes toks faktas jau yra.
    4. Taisyklė "R4 : D -> M" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, L, K, D, M}
--------------------
Iteracija nr. 5:
    1. Taisyklė "R1 : A -> L" netaikoma, nes pažymėta 'flag1'.
    2. Taisyklė "R2 : L -> K" netaikoma, nes pažymėta 'flag1'.
    3. Taisyklė "R3 : D -> A" netaikoma, nes toks faktas jau yra.
    4. Taisyklė "R4 : D -> M" netaikoma, nes pažymėta 'flag1'.
    5. Taisyklė "R5 : F, B -> Z" netaikoma, nes nėra 'prerekvizito' (F) sąraše.
    6. Taisyklė "R6 : C, D -> F" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, L, K, D, M, F}
--------------------
Iteracija nr. 6:
    1. Taisyklė "R1 : A -> L" netaikoma, nes pažymėta 'flag1'.
    2. Taisyklė "R2 : L -> K" netaikoma, nes pažymėta 'flag1'.
    3. Taisyklė "R3 : D -> A" netaikoma, nes toks faktas jau yra.
    4. Taisyklė "R4 : D -> M" netaikoma, nes pažymėta 'flag1'.
    5. Taisyklė "R5 : F, B -> Z" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, L, K, D, M, F, Z}

--------------------------------------------------
3) Rezultatas:
Tikslas 'Z' pasiektas.
Sprendimo planas:
{R1, R2, R7, R4, R6, R5}

Programa baigė darbą.
```

Pavyzdyje pateiktos produkcijų sistemos grafas:



3.2. Čyras vs. Negnevitsky; Čyras laimi.

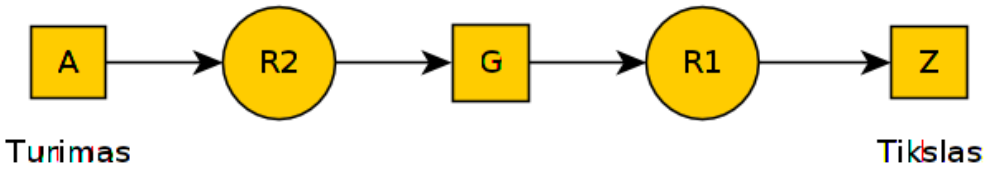
Failo turinys:

```
Pavyzdys 3.
1) Taisyklės:
G Z       // G -> Z
A G       // A -> G
A B       // A -> B
B C       // B -> C
C D       // C -> D
D Z       // D -> Z

2) Faktai:
A

3) Tikslas:
Z
```

Programos rezultatas :

```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

Įvestos taisyklės:
    R1 : G -> Z
    R2 : A -> G
    R3 : A -> B
    R4 : B -> C
    R5 : C -> D
    R6 : D -> Z
Pradiniai faktai: {A}
Tikslas: Z
--------------------------------------------------
2) Sprendimas:

Iteracija nr. 1:
    1. Taisyklė "R1 : G -> Z" netaikoma, nes nėra 'prerekvizito' (G) sąraše.
    2. Taisyklė "R2 : A -> G" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, G}
--------------------
Iteracija nr. 2:
    1. Taisyklė "R1 : G -> Z" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, G, Z}

--------------------------------------------------
3) Rezultatas:
Tikslas 'Z' pasiektas.
Sprendimo planas:
{R2, R1}

Programa baigė darbą.
```

Pavyzdyje pateiktos produkcijų sistemos grafas:



3.3. Čyras vs. Negnevitsky; Negnevitsky laimi su atliekama taisykle.

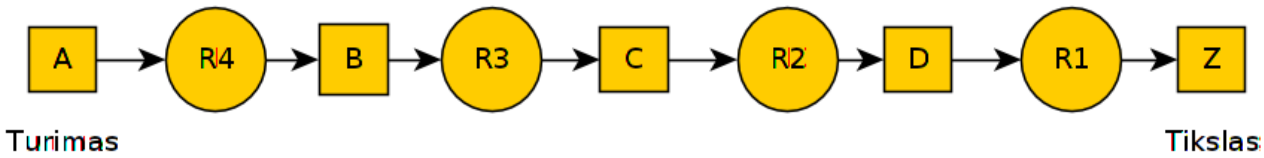
Failo turinys :

```
Pavyzdys 4.
1) Taisyklės:
D Z       // D -> Z
C D       // C -> D
B C       // B -> C
A B       // A -> B
A G       // A -> G
G Z       // G -> Z

2) Faktai:
A

3) Tikslas:
Z
```

Programos rezultatas :

```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

Įvestos taisyklės:
    R1 : D -> Z
    R2 : C -> D
    R3 : B -> C
    R4 : A -> B
    R5 : A -> G
    R6 : G -> Z
Pradiniai faktai: {A}
Tikslas: Z
-------------------------------------------------
2) Sprendimas:

Iteracija nr. 1:
    1. Taisyklė "R1 : D -> Z" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    2. Taisyklė "R2 : C -> D" netaikoma, nes nėra 'prerekvizito' (C) sąraše.
    3. Taisyklė "R3 : B -> C" netaikoma, nes nėra 'prerekvizito' (B) sąraše.
    4. Taisyklė "R4 : A -> B" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B}
--------------------
Iteracija nr. 2:
    1. Taisyklė "R1 : D -> Z" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    2. Taisyklė "R2 : C -> D" netaikoma, nes nėra 'prerekvizito' (C) sąraše.
    3. Taisyklė "R3 : B -> C" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C}
--------------------
Iteracija nr. 3:
    1. Taisyklė "R1 : D -> Z" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    2. Taisyklė "R2 : C -> D" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, D}
--------------------
Iteracija nr. 4:
    1. Taisyklė "R1 : D -> Z" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, D, Z}

-------------------------------------------------
3) Rezultatas:
Tikslas 'Z' pasiektas.
Sprendimo planas:
{R4, R3, R2, R1}

Programa baigė darbą.
```

Pavyzdyje pateiktos produkcijų sistemos grafas:



3.4. Tikslas tarp faktų.

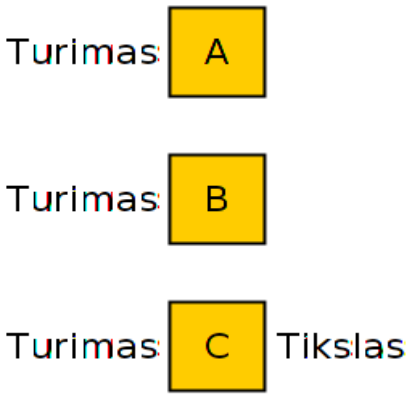
Failo turinys:

```
Pavyzdys 6.
1) Taisyklės:
F B Z          // F, B -> Z
C D F          // C, D -> F
A D       // A -> D

2) Faktai:
A B C

3) Tikslas:
C
```

Programos rezultatas:

```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

Įvestos taisyklės:
    R1 : F, B -> Z
    R2 : C, D -> F
    R3 : A -> D
Pradiniai faktai: {A, B, C}
Tikslas: C
-------------------------------------------------
2) Sprendimas:


-------------------------------------------------
3) Rezultatas:
Tikslas 'C' pasiektas.
Sprendimo planas:
{}

Programa baigė darbą.
```

Pavyzdyje pateiktos produkcijų sistemos grafas:



3.5. Kelias neegzistuoja.

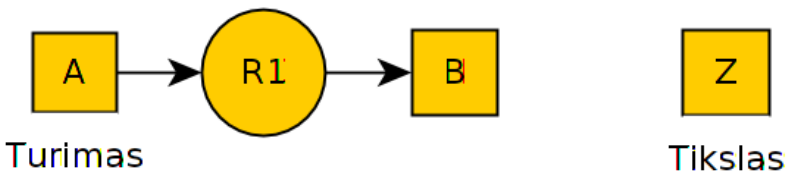
Failo turinys :

```
Pavyzdys 8.
1) Taisyklės:
A B       // R1: A -> B
C Z       // R2: C -> Z

2) Faktai:
A

3) Tikslas:
Z
```

Programos rezultatas :

```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

Įvestos taisyklės:
    R1 : A -> B
    R2 : C -> Z
Pradiniai faktai: {A}
Tikslas: Z
-------------------------------------------------
2) Sprendimas:

Iteracija nr. 1:
    1. Taisyklė "R1 : A -> B" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B}
--------------------
Iteracija nr. 2:
    1. Taisyklė "R1 : A -> B" netaikoma, nes pažymėta 'flag1'.
    2. Taisyklė "R2 : C -> Z" netaikoma, nes nėra 'prerekvizito' (C) sąraše.

-------------------------------------------------
3) Rezultatas:
Tikslas 'Z' nepasiektas.

Programa baigė darbą.
```

Pavyzdyje pateiktos produkcijų sistemos grafas:



3.6. Šeštasis tiesioginio išvedimo pavyzdys

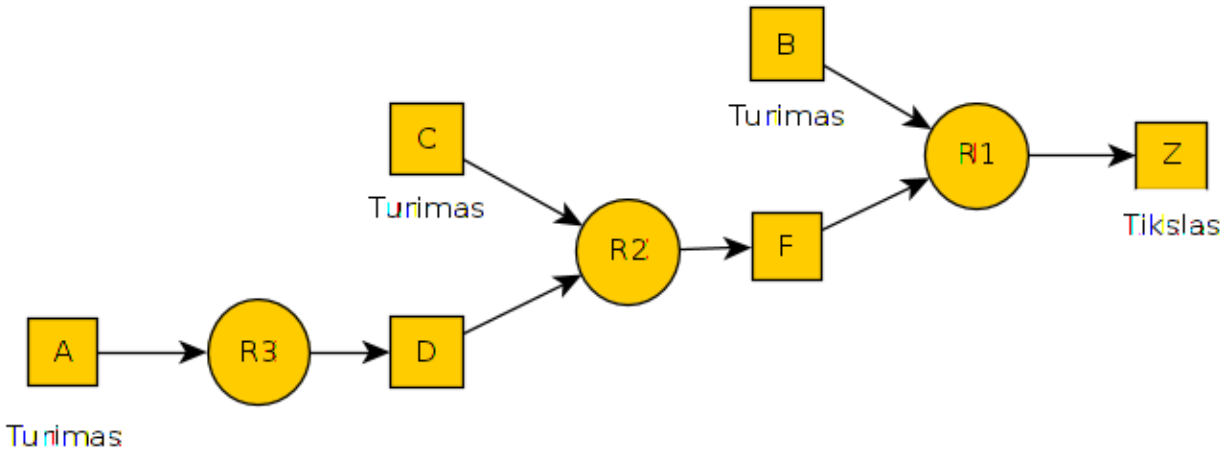
Failo turinys:

```
Pavyzdys 1.
1) Taisyklės:
F B Z          // R1: F, B -> Z
C D F          // R2: C, D -> F
A D            // R3: A -> D

2) Faktai:
A B C

3) Tikslas:
Z
```

Programos rezultatas:

```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

Įvestos taisyklės:
    R1 : F, B -> Z
    R2 : C, D -> F
    R3 : A -> D
Pradiniai faktai: {A, B, C}
Tikslas: Z
-------------------------------------------------
2) Sprendimas:

Iteracija nr. 1:
    1. Taisyklė "R1 : F, B -> Z" netaikoma, nes nėra 'prerekvizito' (F) sąraše.
    2. Taisyklė "R2 : C, D -> F" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    3. Taisyklė "R3 : A -> D" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, D}
--------------------
Iteracija nr. 2:
    1. Taisyklė "R1 : F, B -> Z" netaikoma, nes nėra 'prerekvizito' (F) sąraše.
    2. Taisyklė "R2 : C, D -> F" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, D, F}
--------------------
Iteracija nr. 3:
    1. Taisyklė "R1 : F, B -> Z" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, D, F, Z}

-------------------------------------------------
3) Rezultatas:
Tikslas 'Z' pasiektas.
Sprendimo planas:
{R3, R2, R1}

Programa baigė darbą.
```

Pavyzdyje pateiktos produkcijų sistemos grafas:



3.7. Septintasis tiesioginio išvedimo pavyzdys

Šiame pavyzdyje parodoma, kaip netaikomos produkcijos, kada išeigos faktas (išvada) jau priklauso žinomų faktų aibei. Taip pat šiame pavyzdyje programa nepasiekia terminalinės būsenos, t.y. neranda sprendinio.

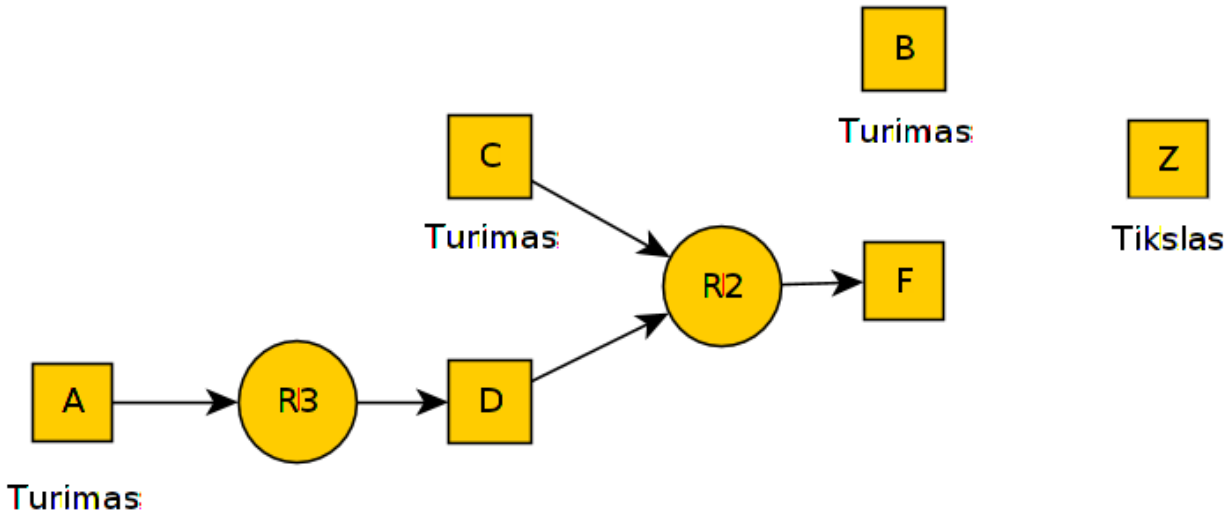
Failo turinys:

```
Pavyzdys 2.
1) Taisyklės:
F B C          // F, B -> C
C D F          // C, D -> F
A D            // A -> D

2) Faktai:
A B C

3) Tikslas:
Z
```

Programos rezultatas:

```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

Įvestos taisyklės:
    R1 : F, B -> C
    R2 : C, D -> F
    R3 : A -> D
Pradiniai faktai: {A, B, C}
Tikslas: Z
-------------------------------------------------
2) Sprendimas:

Iteracija nr. 1:
    1. Taisyklė "R1 : F, B -> C" netaikoma, nes nėra 'prerekvizito' (F) sąraše.
    2. Taisyklė "R2 : C, D -> F" netaikoma, nes nėra 'prerekvizito' (D) sąraše.
    3. Taisyklė "R3 : A -> D" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, D}
--------------------
Iteracija nr. 2:
    1. Taisyklė "R1 : F, B -> C" netaikoma, nes nėra 'prerekvizito' (F) sąraše.
    2. Taisyklė "R2 : C, D -> F" taikoma ir pažymima 'flag1'. Faktų aibė po
pritaikymo: {A, B, C, D, F}
--------------------
Iteracija nr. 3:
    1. Taisyklė "R1 : F, B -> C" netaikoma, nes toks faktas jau yra.
    2. Taisyklė "R2 : C, D -> F" netaikoma, nes pažymėta 'flag1'.
    3. Taisyklė "R3 : A -> D" netaikoma, nes pažymėta 'flag1'.

-------------------------------------------------
3) Rezultatas:
Tikslas 'Z' nepasiektas.

Programa baigė darbą.
```

Pavyzdyje pateiktos produkcijų sistemos grafas:



4. Programos tekstas

Programos tekste vien skaičiumi pavaizduoti komentarai yra programos teksto eilučių išnašos į 2-ajame skyriuje pateikto pseudokodo atitinkamas eilutes.

```
#!/usr/bin/perl

use strict;
use warnings;
no warnings "experimental::smartmatch";
# operatorius 'smartmatch' ('~~') naudojamas elemento buvimui masyve patikrinti

print "Programa pradeda darbą." .
    " [Autorius: Robertas Stankevič, bioinformatikos 4 k.]\n\n";

# Kviečiamas failo nuskaitymas
# Sudaromos duomenų struktūros:
#   1) productions : array of production
#      production :
#        hash : begin
#          premises   => array of char
#          conclusion => char
#          flag => byte
#        end
#   2) assertion_list : array of char
#   3) goal : char
my( $productions, $assertion_list, $goal ) = nuskaityti_is_failo();

# Atspausdinamas failo turinys
print "1) Pateikti pradiniai duomenys:\n\n";
spausdinti_turini( $productions, $assertion_list, $goal );
print "-" x 50 . "\n";

print "2) Sprendimas:\n\n";

my @result;
my $move_number = 1;
my $R = 0;
# { 1}
my $skyrimo_linija = 0;
while( $R < @$productions
       && ! ( $goal ~~ @$assertion_list ) )                                  # { 2}
{                                                                             # { 3}
  if( $R == 0 ){
        $skyrimo_linija ++ and print "-" x 20 . "\n";
        print "Iteracija nr. " . $move_number . ":\n";
    }
  print " " x 4 . ($R + 1) . "." . " " . "Taisyklė "
        . '"' . produkcijos_aprasas( $productions->[$R], $R + 1 ) . '"';

  if( $productions->[$R]->{'flag'} ){
        print " netaikoma, nes pažymėta 'flag1'.\n";
        $R++;
        next;
    }

  if( ar_posarasis( $assertion_list, $productions->[$R]->{'premises'} ) )   # { 4}
    {                                                                       # { 5}
      if( not $productions->[$R]->{'conclusion'} ~~ @$assertion_list )      # { 6}
        {                                                                   # { 7}
          push( @$assertion_list,
                $productions->[$R]->{'conclusion'} );                       # { 8}
```

```
          print " taikoma ir pažymima 'flag1'.";
          print " " x 1 . "Faktų aibė po pritaikymo: {"
            . join( ", ", @$assertion_list ) . "}\n";
          push( @result, "R" . ( $R + 1 ) );
          $productions->[$R]->{'flag'} = 1;
          $R = 0; $move_number ++;                                          # { 9}
        }                                                                   # {10}
        else {
            print " netaikoma, nes toks faktas jau yra.\n";
            $R++;                                                           # {11}
        }
    }                                                                       # {12}
    else {
        printf " netaikoma, nes nėra 'prerekvizito' (%s) sąraše.\n",
            kurio_nera( $assertion_list, $productions->[$R]->{'premises'} );
        $R++;                                                               # {13}
    }
}                                                                           # {14}

print "\n";
print "-" x 50 . "\n";
# Jei išvestų faktų sąraše esama tikslo, pranešama,
# jog tikslas pasiektas
print "3) Rezultatas:\n";
if( $goal ~~ @$assertion_list ) {
  print "Tikslas \'$goal\' pasiektas.\nSprendimo planas:\n"
    . "{" . join( ", ", @result ) . "}" . "\n";
} else {
  print "Tikslas \'$goal\' nepasiektas.\n";
}
print "\n";
print "Programa baigė darbą.\n";

sub nuskaityti_is_failo
{
    my( @productions, @assertion_list, $goal );

    <>, <>;                   # Praleidžiamos pirmos dvi failo eilutės

    while( <> ) {             # Po vieną skaitomos tolimesnės įvesties failo eilutės
        last if $_ =~ /^$/;   # Ciklas paliekamas, jei sutinkama tuščia eilutė

        # Perskaitytos eilutės pabaigoje ištrinamas komentaras (jeigu yra).
        $_ =~ s{ //.* }{}x;
        # Perskaityta eilutė apkarpoma (trim) ir skaldoma pagal tarpų simbolius,
        # pagaminant sąrašą iš simbolių
        my @production = split ' ', $_;
        # Paskutinis perskaitytos eilutės simbolis yra išvada
        my $conclusion = pop @production;
        # Į masyvo pabaigą pridedamas įrašas, atitinkantis produkciją
        push( @productions,
            { 'premises' => \@production, 'conclusion' => $conclusion, 'flag' => 0 }
        );
    }

    <>;                        # Praleidžiama sekanti failo eilutė

    # Kita failo eilutė apkarpoma (trim) ir skaldoma pagal tarpų simbolius,
    # pagaminant masyvą iš simbolių
    @assertion_list = split ' ', <>;

    <>, <>;                          # Praleidžiamos dvi sekančios failo eilutės
    <> =~ /./ or warn "Ši eilutė neturi būti be simbolių!\n";

    # Kitos failo eilutės pirmas simbolis priskiriamas tikslui
    $goal = $&;
    return( \@productions, \@assertion_list, $goal );
```

```
}

# Procedūra, struktūrizuotai spausdinanti failo turinį
sub spausdinti_turini
{
    my( $productions, $assertion_list, $goal ) = @_;
    print "Įvestos taisyklės:\n";

    # Atspausdinama kiekviena produkcija
    for( my $i = 0; $i < @$productions; $i++ ) {
        print " " x 4
          . produkcijos_aprasas( $productions->[$i], $i + 1 ) . "\n";
    }

    # Atspausdinami faktai, jų pavadinimus atskyrus kableliu
    print "Pradiniai faktai: {"
      . join( ", ", @$assertion_list ) . "}\n";
    print "Tikslas: " . $goal . "\n";
}

# Funkcija, grąžinanti tekstinę produkcijos reprezentaciją
# "Produkcijos vardas: prerekvizitai, atskirti kableliais -> išvada"
sub produkcijos_aprasas
{
    my( $production, $number ) = @_;
    return "R" . $number . " : "
        . join( ", ", @{ $production->{'premises'} } )
        . " -> " . $production->{'conclusion'};
}

# Funkcija, tikrinanti, ar visi vieno sąrašo elementai
# yra kitame sąraše
sub ar_posarasis
{
    my( $list, $sublist ) = @_;

    # Kintamajam $_ priskiriamas kaskart vis kitas sąrašo elementas
    foreach( @$sublist ) {

        # Jei elemento $_ nėra sąraše, grąžinama 0
        return 0 if not $_ ~~ @$list;
    }
    return 1;  # Suradus visus elementus, grąžinama 1
}

# Funkcija, grąžinanti, kurio iš vieno sąrašo elementų
# nėra kitame sąraše
sub kurio_nera
{
    my( $list, $sublist ) = @_;

    # Kintamajam $_ priskiriamas kaskart vis kitas sąrašo elementas
    foreach( @$sublist ) {

        # Jei elemento $_ nėra sąraše, jis grąžinamas
        return $_ if not $_ ~~ @$list;
    }
    warn "Elementas turėjo būti surastas!\n";  # Pranešimas apie klaidą
}
```

5. Literatūra

1. Straipsnis apie tiesioginį išvedimą Wikipedia.org *http://en.wikipedia.org/wiki/Forward_chaining*

2. V. Čyras. Intelektualios sistemos. *http://www.mif.vu.lt/~cyras/konspektas-_intelektualios-sistemos.pdf*, 122.