VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATINĖS INFORMATIKOS KATEDRA

Robertas Stankevič
Bioinformatikos studijų programos
IV kurso studentas

Dirbtinis intelektas.
Laboratorinis darbas Nr. 2.
Atbulinio išvedimo (*Backward chaining*) programa
Perl kalba

Vadovas Vytautas Čyras

Vilnius 2015

Turinys

1. Įvadas

Atbulinis išvedimas (angl. *Backward 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 (globalią duomenų bazę) iš pradinės būsenos pervestų į terminalinę būseną, arba išsiaiškintų, kad terminalinė būsena nepasiekiama.
Atbulinio išvedimo užduotis yra surasti faktus, prieš tai žinant tikslą.
Vykdomos iteracijos, ir kiekvienoje paeiliui ieškoma produkcijos iš produkcijų sąrašo, kurią galima būtų pritaikyti, t.y. kurios dešinioji pusė (konsekventas, kitaip - išvada) atitiktų ieškomą tikslą. Procedūra yra rekursyvi. Suradus produkciją, atitikančią tikslą, jos antecedentai (faktai kairiojoje pusėje) tampa naujais tikslais. Rekursija vykdoma tol, kol naujais tikslais yra pasiekiami visi pradiniai GDB faktai. Pritaikytų taisyklių (produkcijų) seka sudaro teigiamą uždavinio atsakymą. Neigiamas uždavinio atsakymas gaunamas tada, kada mėgintos pritaikyti visos taisyklės, ir gautieji nauji tikslai nepadengia pradinių duotųjų faktų. Reiškia, kad tikslo pasiekti, turint šią pradinę GDB ir šias produkcijas, neįmanoma.
Laboratorinio darbo tikslas buvo parašyti atbulinio 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, įvairių pavyzdžių įvesčių ir išvesčių (įskaitant verifikavimo grafą pseudografiniu pavidalu), semantinių grafų schemas dvidalio grafo pavidalu.

2. Algoritmas

Algoritmo įvestis:

1. produkcijų sąrašas;
2. pradinių faktų sąrašas;
3. tikslas.

Algoritmo išvestis:

1. atsakymas, ar tikslas pasiekiamas.
2. paeiliui naudojamų produkcijų sąrašas (planas).

2.1. Algoritmo žingsniai

1. Jei duotas tikslas patenka į visų tikslų sąrašą, funkcija grąžina, false reikšmę (pranešimas apie ciklą).
2. Jei duotas tikslas patenka į pradinių faktų sąrašą, funkcija grąžina, kad duotas tikslas pasiektas.
3. Jei duotas tikslas patenka į gautųjų faktų aibę, funkcija grąžina, kad duotas tikslas pasiektas.
4. Tikslas pridedamas prie visų tikslų sąrašo.
5. Einama per taisyklių sąrašą (imama sekanti taisyklė iš sąrašo), jei pasiekta sąrašo pabaiga einama į 14 žingsnį.
6. Jei taisyklės konsekventas nėra duotasis tikslas, einama į 5 žingsnį.
7. Išsaugomi gautųjų faktų ir produkcijų sąrašų dydžiai.
8. Pažymima, kad einamoji taisyklė yra taikytina.
9. Einama per taisyklės kairės pusės faktų sąrašą: kiekvienam faktui rekursiškai kviečiama funkcija pateikiant faktą iš taisyklės kaip duotą tikslą.
10. Jei bent vienam faktui funkcija grąžina, kad nepavyko rasti duoto tikslo, išvestų faktų ir produkcijų sąrašai grąžinami į prieš rekursiją išsaugotas būsenas ir einama į 5 žingsnį.
11. Taisyklės konsekventas pridedamas prie gautųjų faktų sąrašo.
12. Taisyklė pridedama prie produkcijų sąrašo.
13. Duotas tikslas ištrinamas iš visų tikslų sąrašo ir funkcija grąžina, kad duotas tikslas pasiektas.
14. Funkcija grąžina, kad duotas tikslas nepasiektas.

2.1. Algoritmo pseudokodas

```
boolean backward_chaining( tikslas ) {
    IF (tikslas patenka į visų tikslų sąrašą){                              //1
        RETURN FALSE;
    }
    IF (tikslas patenka į pradinių faktų sąrašą) {                          //2
        RETURN TRUE;
    }
    IF (tikslas patenka į gautųjų faktų aibę) {                             //3
        RETURN TRUE;
    }
    tikslas pridedamas prie visų tikslų sąrašo;                             //4
    FOREACH (kiekvienai taisyklei iš taisyklių sąrašo) {                    //5
        IF (taisyklės konsekventas sutampa su duotu tikslu){                //6
        išsaugomi gautųjų faktų ir produkcijų sąrašų dydžiai                //7
        taisyklė_tinkama_taikyti = TRUE;                                    //8
        FOREACH (kiekvienam taisyklės kairės pusės faktui) {                //9
            IF (backward_chaining( taisyklės kairės pusės faktas )) {       //10
                gautųjų faktų ir produkcijų sąrašai grąžinami į prieš
                    rekursiją išsaugotas būsenas;
                taisyklė_tinkama_taikyti = FALSE;
                BREAK;
            }
        }
        IF (taisyklė_tinkama_taikyti) {
            taisyklės dešinioji pusė pridedama prie gautųjų faktų sąrašo;   //11
            taisyklė pridedama prie produkcijų sąrašo;                      //12
            tikslas ištrinamas iš visų tikslų sąrašo;                       //13
            RETURN TRUE;
        }
    }
    RETURN FALSE;                                                           //14
}
```

3. Duomenų įvesties failo formatas

```
1 testas.
----------------------
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
```

4. Pavyzdžiai

4.1. Pirmasis pavyzdys

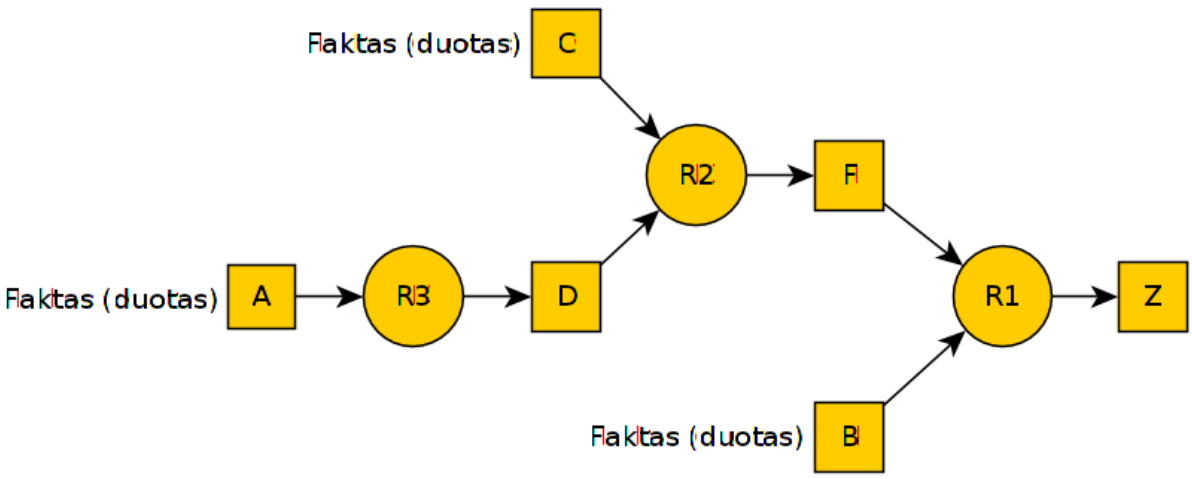
Failo turinys:

```
1 testas.
---------------------
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:

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

  1. Tikslas Z. Randame R1 : F, B -> Z. Nauji tikslai: F, B
  2.   Tikslas F. Randame R2 : C, D -> F. Nauji tikslai: C, D
  3.     Tikslas C. Faktas (duotas).
  4.     Tikslas D. Randame R3 : A -> D. Nauji tikslai: A
  5.       Tikslas A. Faktas (duotas).
  6.     Tikslas D. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D}.
  7.   Tikslas F. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D, F}.
  8.   Tikslas B. Faktas (duotas).
  9. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D, F, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R3, R2, R1]
Verifikavimo grafas:

[A B C] -R3→ [A B C D] -R2→ [A B C D F] -R1→ [A B C D F Z]

Programa baigė darbą.
```

Semantinis grafas:



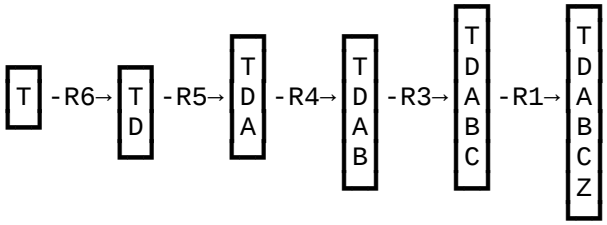
4.2. Antrasis pavyzdys (konspektų 1-as testas)

Failo turinys:

```
2 testas.
----------------------
1) Taisyklės:
D C Z          // R1: D, C -> Z
C D       // R2: C -> D
B C       // R3: B -> C
A B       // R4: A -> B
D A       // R5: D -> A
T D       // R6: T -> D
G A       // R7: G -> A
H B       // R8: H -> B
J C       // R9: J -> C
----------------------
2) Faktai:
T
----------------------
3) Tikslas:
Z
```

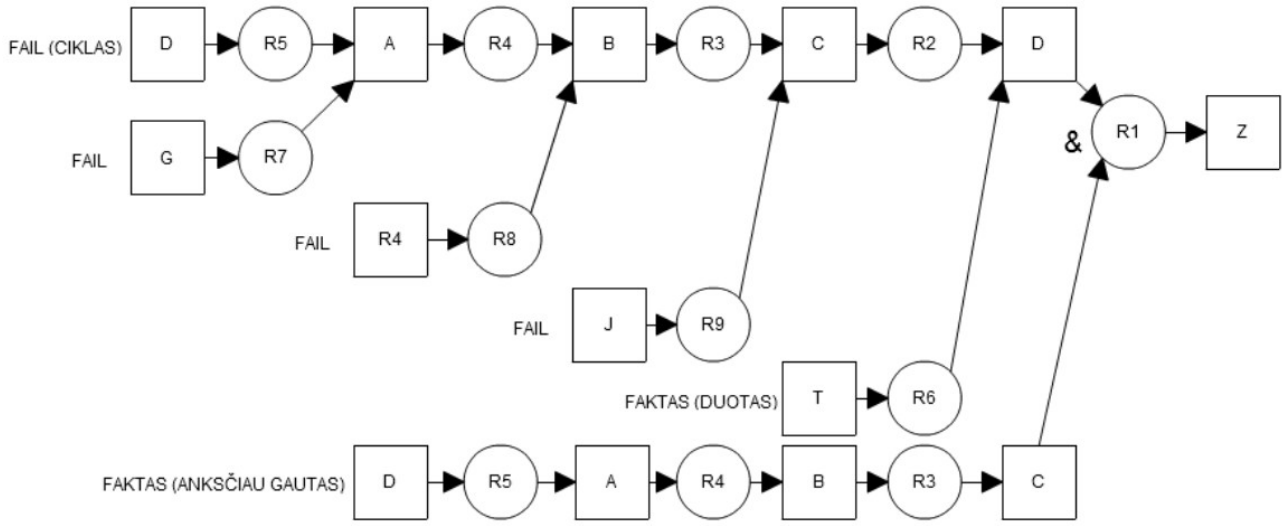
Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : D, C -> Z
       R2 : C -> D
       R3 : B -> C
       R4 : A -> B
       R5 : D -> A
       R6 : T -> D
       R7 : G -> A
       R8 : H -> B
       R9 : J -> C
  1.2) Pradiniai faktai: {T}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R1 : D, C -> Z. Nauji tikslai: D, C
  2.   Tikslas D. Randame R2 : C -> D. Nauji tikslai: C
  3.     Tikslas C. Randame R3 : B -> C. Nauji tikslai: B
  4.       Tikslas B. Randame R4 : A -> B. Nauji tikslai: A
  5.         Tikslas A. Randame R5 : D -> A. Nauji tikslai: D
  6.           Tikslas D. FAIL - ciklas.
  7.         Tikslas A. Randame R7 : G -> A. Nauji tikslai: G
  8.           Tikslas G. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
  9.         Tikslas A. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
 10.       Tikslas B. Randame R8 : H -> B. Nauji tikslai: H
 11.         Tikslas H. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
 12.       Tikslas B. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
 13.     Tikslas C. Randame R9 : J -> C. Nauji tikslai: J
 14.       Tikslas J. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
 15.     Tikslas C. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
 16.   Tikslas D. Randame R6 : T -> D. Nauji tikslai: T
 17.     Tikslas T. Faktas (duotas).
 18.   Tikslas D. Faktas (dabar naujai gautas) ir GDB = {T, D}.
 19.   Tikslas C. Randame R3 : B -> C. Nauji tikslai: B
 20.     Tikslas B. Randame R4 : A -> B. Nauji tikslai: A
 21.       Tikslas A. Randame R5 : D -> A. Nauji tikslai: D
 22.         Tikslas D. Faktas (buvo naujai gautas).
 23.       Tikslas A. Faktas (dabar naujai gautas) ir GDB = {T, D, A}.
 24.     Tikslas B. Faktas (dabar naujai gautas) ir GDB = {T, D, A, B}.
 25.   Tikslas C. Faktas (dabar naujai gautas) ir GDB = {T, D, A, B, C}.
 26. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {T, D, A, B, C, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R6, R5, R4, R3, R1]
Verifikavimo grafas:
```




```
Programa baigė darbą.
```

Semantinis grafas:



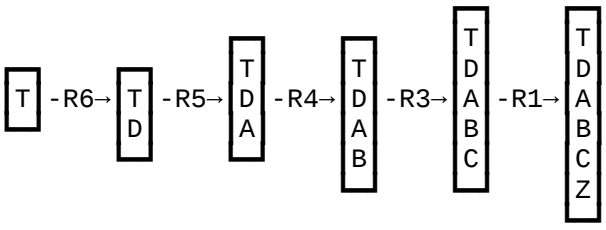
4.3. Trečiasis pavyzdys (konspektų 2-as testas)

Failo turinys:

```
3 testas.
----------------------
1) Taisyklės:
C D Z          // R1: C, D -> Z
C D       // R2: C -> D
B C       // R3: B -> C
A B       // R4: A -> B
D A       // R5: D -> A
T D       // R6: T -> D
G A       // R7: G -> A
H B       // R8: H -> B
J C       // R9: J -> C
----------------------
2) Faktai:
T
----------------------
3) Tikslas:
Z
```

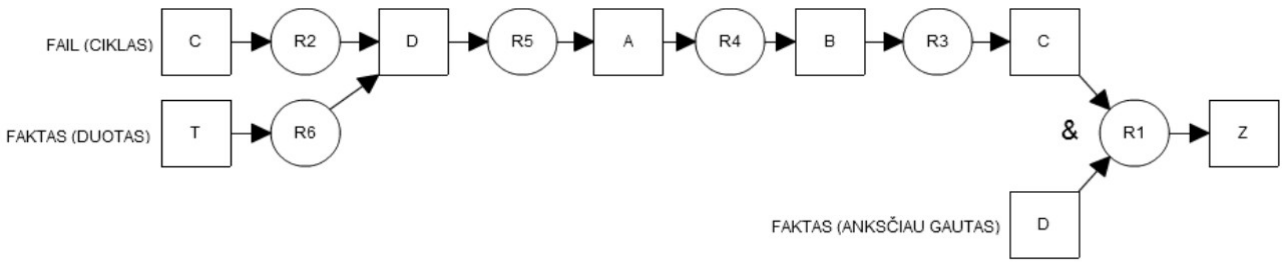
Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : C, D -> Z
       R2 : C -> D
       R3 : B -> C
       R4 : A -> B
       R5 : D -> A
       R6 : T -> D
       R7 : G -> A
       R8 : H -> B
       R9 : J -> C
  1.2) Pradiniai faktai: {T}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R1 : C, D -> Z. Nauji tikslai: C, D
  2.   Tikslas C. Randame R3 : B -> C. Nauji tikslai: B
  3.     Tikslas B. Randame R4 : A -> B. Nauji tikslai: A
  4.       Tikslas A. Randame R5 : D -> A. Nauji tikslai: D
  5.         Tikslas D. Randame R2 : C -> D. Nauji tikslai: C
  6.           Tikslas C. FAIL - ciklas.
  7.         Tikslas D. Randame R6 : T -> D. Nauji tikslai: T
  8.           Tikslas T. Faktas (duotas).
  9.         Tikslas D. Faktas (dabar naujai gautas) ir GDB = {T, D}.
 10.       Tikslas A. Faktas (dabar naujai gautas) ir GDB = {T, D, A}.
 11.     Tikslas B. Faktas (dabar naujai gautas) ir GDB = {T, D, A, B}.
 12.   Tikslas C. Faktas (dabar naujai gautas) ir GDB = {T, D, A, B, C}.
 13.   Tikslas D. Faktas (buvo naujai gautas).
 14. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {T, D, A, B, C, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R6, R5, R4, R3, R1]
Verifikavimo grafas:
```



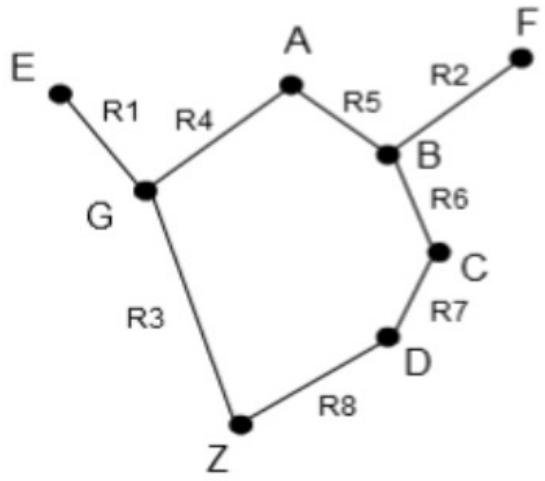

```
Programa baigė darbą.
```

Semantinis grafas:



4.4. Ketvirtasis pavyzdys (konspektų 5-as testas)

Duotas grafas:



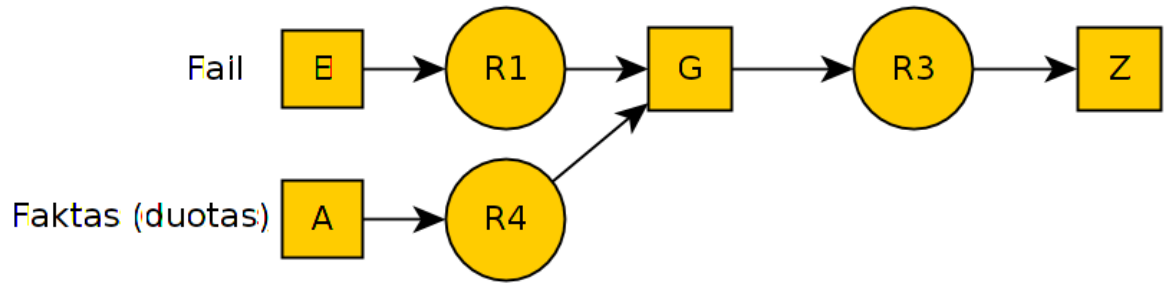
Failo turinys:

```
4 testas.
----------------------
1) Taisyklės:
E G       // R1: E -> G
F B       // R2: F -> B
G Z       // R3: G -> Z
A G       // R4: A -> G
A B       // R5: A -> B
B C       // R6: B -> C
C D       // R7: C -> D
D Z       // R8: D -> Z
----------------------
2) Faktai:
A
----------------------
3) Tikslas:
Z
```

Programos rezultatas:

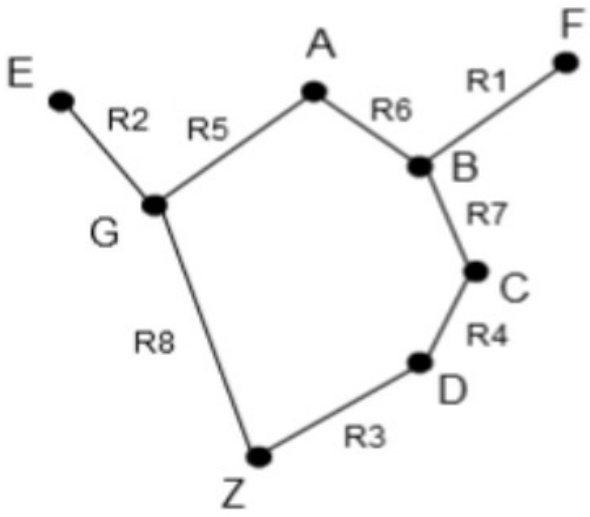
```
Programa pradeda darbą. [Autorius: Robertas Stankevič, bioinformatikos 4 k.]

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : E -> G
       R2 : F -> B
       R3 : G -> Z
       R4 : A -> G
       R5 : A -> B
       R6 : B -> C
       R7 : C -> D
       R8 : D -> Z
  1.2) Pradiniai faktai: {A}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R3 : G -> Z. Nauji tikslai: G
  2.   Tikslas G. Randame R1 : E -> G. Nauji tikslai: E
  3.     Tikslas E. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
  4.   Tikslas G. Randame R4 : A -> G. Nauji tikslai: A
  5.     Tikslas A. Faktas (duotas).
  6.   Tikslas G. Faktas (dabar naujai gautas) ir GDB = {A, G}.
  7. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {A, G, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R4, R3]
Verifikavimo grafas:

[A] -R4→ [A G] -R3→ [A G Z]

Programa baigė darbą.
```

Semantinis grafas:



4.5. Penktasis pavyzdys

Duotas grafas:



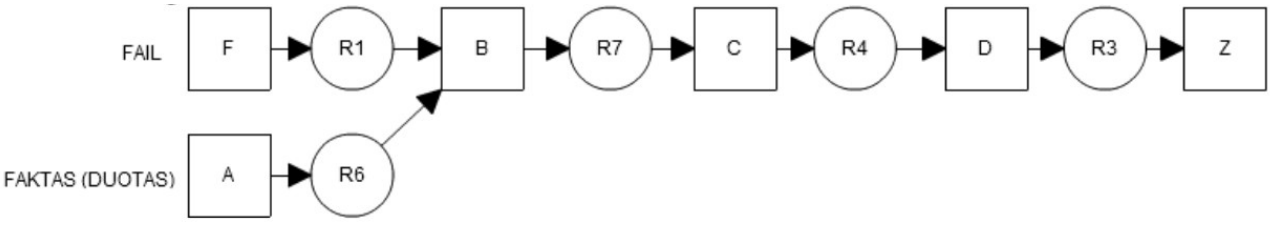
Failo turinys:

```
5 testas.
----------------------
1) Taisyklės:
F B       // R1: F -> B
E G       // R2: E -> G
D Z       // R3: D -> Z
C D       // R4: C -> D
A G       // R5: A -> G
A B       // R6: A -> B
B C       // R7: B -> C
G Z       // R8: G -> Z
----------------------
2) Faktai:
A
----------------------
3) Tikslas:
Z
```

Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : F -> B
       R2 : E -> G
       R3 : D -> Z
       R4 : C -> D
       R5 : A -> G
       R6 : A -> B
       R7 : B -> C
       R8 : G -> Z
  1.2) Pradiniai faktai: {A}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R3 : D -> Z. Nauji tikslai: D
  2.   Tikslas D. Randame R4 : C -> D. Nauji tikslai: C
  3.     Tikslas C. Randame R7 : B -> C. Nauji tikslai: B
  4.       Tikslas B. Randame R1 : F -> B. Nauji tikslai: F
  5.         Tikslas F. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
  6.       Tikslas B. Randame R6 : A -> B. Nauji tikslai: A
  7.         Tikslas A. Faktas (duotas).
  8.       Tikslas B. Faktas (dabar naujai gautas) ir GDB = {A, B}.
  9.     Tikslas C. Faktas (dabar naujai gautas) ir GDB = {A, B, C}.
 10.   Tikslas D. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D}.
 11. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R6, R7, R4, R3]
Verifikavimo grafas:

[A] -R6→ [A B] -R7→ [A B C] -R4→ [A B C D] -R3→ [A B C D Z]

Programa baigė darbą.
```

Semantinis grafas:



4.6. Šeštasis pavyzdys (konspektų 3-as testas)

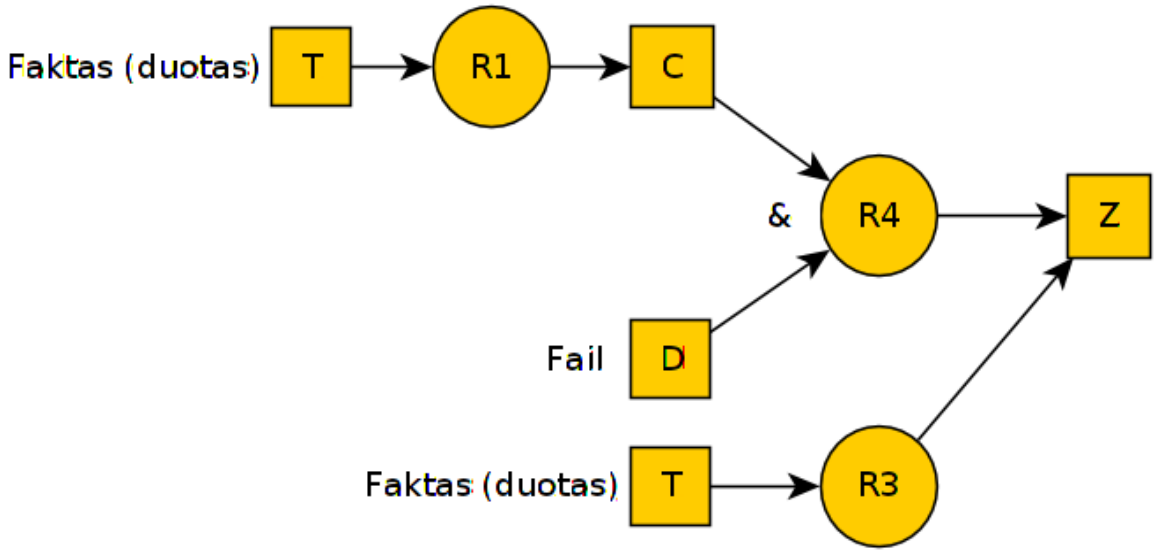
Failo turinys:

```
6 testas.
----------------------
1) Taisyklės:
C D Z          // R1: C, D -> Z
T C       // R2: T -> C
T Z       // R3: T -> Z
----------------------
2) Faktai:
T
----------------------
3) Tikslas:
Z
```

Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : C, D -> Z
       R2 : T -> C
       R3 : T -> Z
  1.2) Pradiniai faktai: {T}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R1 : C, D -> Z. Nauji tikslai: C, D
  2.   Tikslas C. Randame R2 : T -> C. Nauji tikslai: T
  3.     Tikslas T. Faktas (duotas).
  4.   Tikslas C. Faktas (dabar naujai gautas) ir GDB = {T, C}.
  5.   Tikslas D. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
  6. Tikslas Z. Randame R3 : T -> Z. Nauji tikslai: T
  7.   Tikslas T. Faktas (duotas).
  8. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {T, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R3]
Verifikavimo grafas:
┌─┐      ┌─┐
│T│ -R3→ │T│
└─┘      │Z│
         └─┘
Programa baigė darbą.
```

Semantinis grafas:



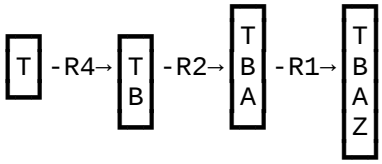
4.7. Septintasis pavyzdys (konspektų 4-as testas)

Failo turinys:

```
7 testas.
---------------------
1) Taisyklės:
A Z       // R1: A -> Z
B A       // R2: B -> A
A C B          // R3: A, C -> B
T B       // R4: T -> B
T C       // R5: T -> C
---------------------
2) Faktai:
T
---------------------
3) Tikslas:
Z
```

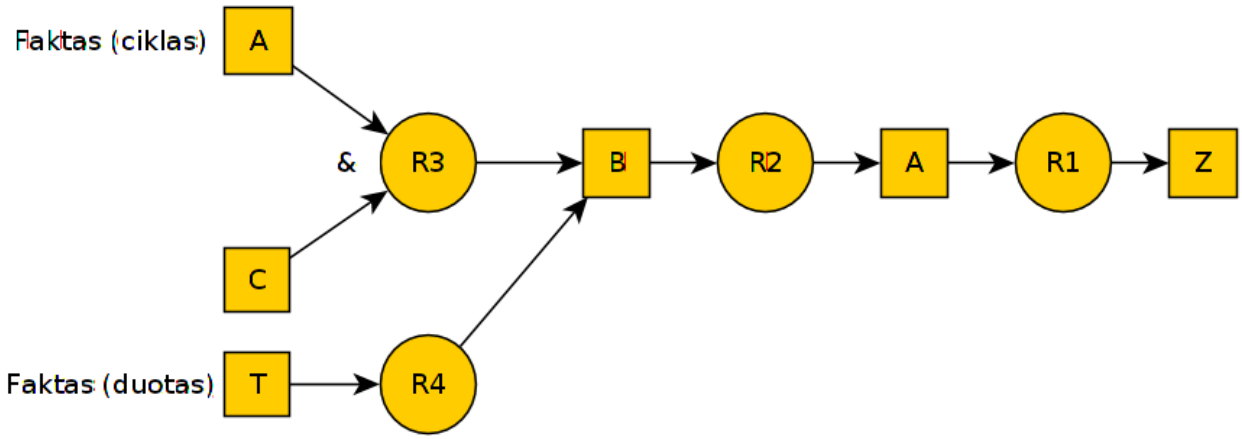
Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : A -> Z
       R2 : B -> A
       R3 : A, C -> B
       R4 : T -> B
       R5 : T -> C
  1.2) Pradiniai faktai: {T}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R1 : A -> Z. Nauji tikslai: A
  2.   Tikslas A. Randame R2 : B -> A. Nauji tikslai: B
  3.     Tikslas B. Randame R3 : A, C -> B. Nauji tikslai: A, C
  4.       Tikslas A. FAIL - ciklas.
  5.     Tikslas B. Randame R4 : T -> B. Nauji tikslai: T
  6.       Tikslas T. Faktas (duotas).
  7.     Tikslas B. Faktas (dabar naujai gautas) ir GDB = {T, B}.
  8.   Tikslas A. Faktas (dabar naujai gautas) ir GDB = {T, B, A}.
  9. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {T, B, A, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R4, R2, R1]
Verifikavimo grafas:
```




```
Programa baigė darbą.
```

Semantinis grafas:



4.8. Aštuntasis pavyzdys

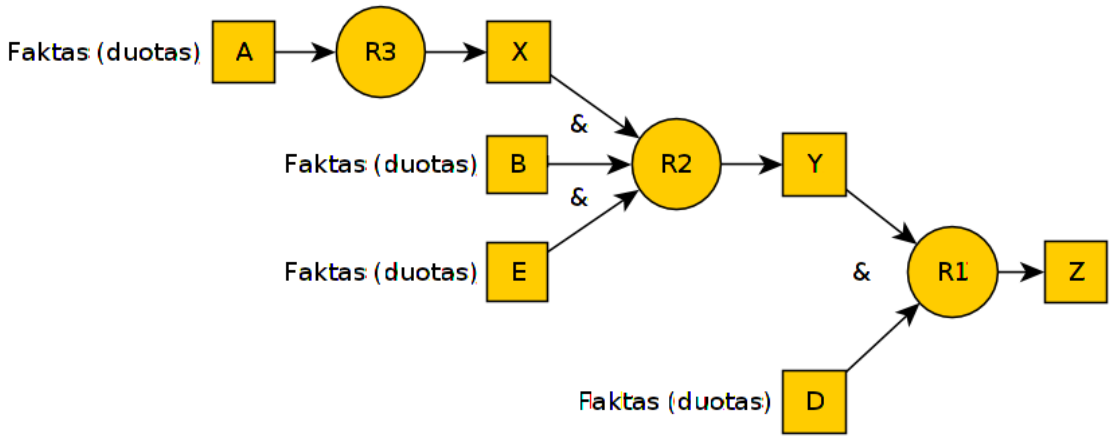
Failo turinys:

```
8 testas.
----------------------
1) Taisyklės:
Y D Z          // R1: Y, D -> Z
X B E Y        // R2: X, B, E -> Y
A X       // R3: A -> X
C L       // R4: C -> L
L M N          // R5: L, M -> N
----------------------
2) Faktai:
A B C D E
----------------------
3) Tikslas:
Z
```

Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : Y, D -> Z
       R2 : X, B, E -> Y
       R3 : A -> X
       R4 : C -> L
       R5 : L, M -> N
  1.2) Pradiniai faktai: {A, B, C, D, E}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R1 : Y, D -> Z. Nauji tikslai: Y, D
  2.   Tikslas Y. Randame R2 : X, B, E -> Y. Nauji tikslai: X, B, E
  3.     Tikslas X. Randame R3 : A -> X. Nauji tikslai: A
  4.       Tikslas A. Faktas (duotas).
  5.     Tikslas X. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D, E, X}.
  6.     Tikslas B. Faktas (duotas).
  7.     Tikslas E. Faktas (duotas).
  8.   Tikslas Y. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D, E, X, Y}.
  9.   Tikslas D. Faktas (duotas).
 10. Tikslas Z. Faktas (dabar naujai gautas) ir GDB = {A, B, C, D, E, X, Y, Z}.
-------------------------------------------------
3) Rezultatas:

Tikslas pasiekiamas.
Planas: [R3, R2, R1]
Verifikavimo grafas:

[A B C D E] -R3→ [A B C D E X] -R2→ [A B C D E X Y] -R1→ [A B C D E X Y Z]

Programa baigė darbą.
```

Semantinis grafas:



4.9. Devintasis pavyzdys (tikslas nepasiekiamas)

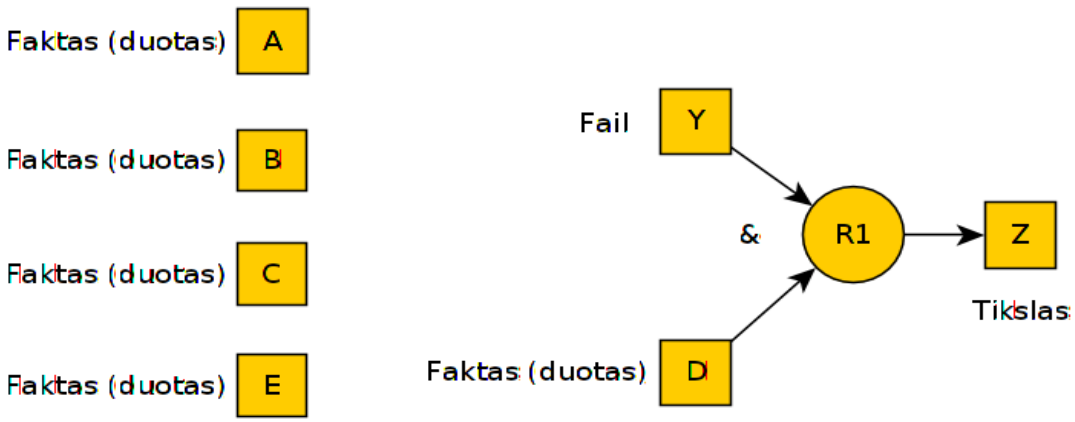
Failo turinys:

```
9 testas.
----------------------
1) Taisyklės:
Y D Z          // R1: Y, D -> Z
----------------------
2) Faktai:
A B C D E
----------------------
3) Tikslas:
Z
```

Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : Y, D -> Z
  1.2) Pradiniai faktai: {A, B, C, D, E}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Randame R1 : Y, D -> Z. Nauji tikslai: Y, D
  2.   Tikslas Y. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
  3. Tikslas Z. FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.
-------------------------------------------------
3) Rezultatas:

Tikslas nepasiekiamas.

Programa baigė darbą.
```

Semantinis grafas:



4.10. Dešimtasis pavyzdys (tikslas faktų aibėje)

Failo turinys:

```
10 testas.
----------------------
1) Taisyklės:
Y U       // R1: Y -> U
----------------------
2) Faktai:
Z F
----------------------
3) Tikslas:
Z
```

Programos rezultatas:

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

1) Pateikti pradiniai duomenys:

  1.1) Įvestos taisyklės:
       R1 : Y -> U
  1.2) Pradiniai faktai: {Z, F}
  1.3) Tikslas: Z
-------------------------------------------------
2) Sprendimas:

  1. Tikslas Z. Faktas (duotas).
-------------------------------------------------
3) Rezultatas:

Tikslas faktų aibėje.

Programa baigė darbą.
```

5. 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) taisykles : ARRAY OF taisykle
#      taisykle :
#        HASH : BEGIN
#          'premises'   => ARRAY OF CHAR
#          'conclusion' => CHAR
#        END
#   2) pradiniai_faktai : ARRAY OF CHAR
#   3) tikslas : CHAR
my( $taisykles, $pradiniai_faktai, $tikslas ) = _nuskaityti_is_failo();

# Atspausdinamas failo turinys
print "1) Pateikti pradiniai duomenys:\n\n";
    _spausdinti_turini( $taisykles, $pradiniai_faktai, $tikslas );

print "-" x 50 . "\n";

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

# Kintamieji - nuorodos į tuščius anoniminius masyvus.
my $tikslai = [];
my $gautieji_faktai = [];
my $taisykliu_seka = [];

# Kviečiama rekursinė atbulinio išvedimo funkcija.
my $BC_ok = _backward_chaining( $tikslas, 0 );

print "-" x 50 . "\n";

# Spausdinamas rezultatas:
print "3) Rezultatas:\n\n"
    . do {
        if( $BC_ok ){
            if( @$taisykliu_seka ){
                "Tikslas pasiekiamas.\n"
                . ( join '', map "Planas: [$_]\n",
                    join ', ', map "R$_", @$taisykliu_seka )
                . _verifikavimo_grafas( $taisykliu_seka,
                    $pradiniai_faktai, $gautieji_faktai );
            }
            else {
                "Tikslas faktų aibėje.\n"
            }
        }
```

```
        else {
            "Tikslas nepasiekiamas.\n"
        }
    };

print "\n";
print "Programa baigė darbą.\n";

# Algoritmo žingsnių skaitliukas.
my $step = 0;

# Atbulinio išvedimo rekursinė funkcija
sub _backward_chaining {
    my ($tikslas, $tab) = @_;

    return undef  if $tikslas ~~ @$tikslai             and print                  # {1}
                                  _sp_tikslas( ++ $step, $tab, $tikslas ),
                                  " FAIL - ciklas.\n";
    return 'true' if $tikslas ~~ @$pradiniai_faktai    and print                  # {2}
                                  _sp_tikslas( ++ $step, $tab, $tikslas ),
                                  " Faktas (duotas).\n";
    return 'true' if $tikslas ~~ @$gautieji_faktai     and print                  # {3}
                                  _sp_tikslas( ++ $step, $tab, $tikslas ),
                                  " Faktas (buvo naujai gautas).\n";
    push @$tikslai, $tikslas;                                                   # {4}

    my $taisykles_nr;
    for my $taisykle ( @$taisykles ){                                           # {5}

        $taisykles_nr ++;

        if( $taisykle->{'conclusion'} eq $tikslas ){                            # {6}

            print _sp_tikslas( ++ $step, $tab, $tikslas );
            printf " Randame %s. Nauji tikslai: %s\n",
                _produkcijos_aprasas( $taisykle, $taisykles_nr ),
                join ', ', @{$taisykle->{'premises'}};

            my $issaugotas_gautuju_faktu_skaicius = @$gautieji_faktai;         # {7.1}
            my $issaugotas_taisykliu_sekoje_skaicius = @$taisykliu_seka;       # {7.2}
            my $naudotina = 'true';                                             # {8}

            for my $faktas ( @{ $taisykle->{'premises'} } ){                    # {9}
                if( not _backward_chaining( $faktas, $tab + 1 ) ){              # {10}
                    splice @$gautieji_faktai,
                        $issaugotas_gautuju_faktu_skaicius;
                    splice @$taisykliu_seka,
                        $issaugotas_taisykliu_sekoje_skaicius;
                    undef $naudotina;
                    last;
                }
            }
            if( $naudotina ){

                # Taisyklės išvada pridedama prie gautųjų faktų sąrašo
                push @$gautieji_faktai, $taisykle->{'conclusion'};              # {11}
                # Taisyklės numeris pridedamas prie rezultato taisyklių sekos
                push @$taisykliu_seka, $taisykles_nr;                           # {12}

                print _sp_tikslas( ++ $step, $tab, $tikslas );
                printf " Faktas (dabar naujai gautas) ir GDB = {%s}.\n",
                    join ", ", @$pradiniai_faktai, @$gautieji_faktai;

                # Tikslas ištrinamas iš visų tikslų sąrašo
                @$tikslai = grep $_ ne $tikslas, @$tikslai;                     # {13}
                return 'true';
            }
```

```
        }
    }

    print _sp_tikslas( ++ $step, $tab, $tikslas );
    print " FAIL, nes daugiau nėra taisyklių išvesti šį tikslą.\n";
    pop @$tikslai;
    return undef;                                                          # {14}
}

# Funkcija, nuskaitanti failo duomenis
sub _nuskaityti_is_failo
{
    my( @taisykles, @pradiniai_faktai, $tikslas );

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

    # Po vieną skaitomos tolimesnės įvesties failo eilutės
    while( <> ) {

        # Ciklas paliekamas, jei sutinkama tuščia arba minusais uzpildyta eilutė
        last if $_ =~ /^-*$/;

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

    <>;                     # Praleidžiama sekanti failo eilutė

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

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

    # Kitos failo eilutės pirmas simbolis priskiriamas tikslui
    $tikslas = $&;
    return( \@taisykles, \@pradiniai_faktai, $tikslas );
}

# Procedūra, struktūrizuotai spausdinanti failo turinį
sub _spausdinti_turini {
    my( $taisykles, $pradiniai_faktai, $tikslas ) = @_;
    print ' ' x 2 . "1.1) Įvestos taisyklės:\n";

    # Atspausdinama kiekviena taisyklė
    for( my $i = 0; $i < @$taisykles; $i++ ) {
        print " " x 8
        . _produkcijos_aprasas( $taisykles->[$i], $i + 1 ) . "\n";
    }

    # Atspausdinami faktai, jų pavadinimus atskyrus kableliu
    print ' ' x 2 . "1.2) Pradiniai faktai: {"
        . join( ", ", @$pradiniai_faktai ) . "}\n";
    print ' ' x 2 . "1.3) Tikslas: " . $tikslas . "\n";
}

# Funkcija, grąžinanti tekstinę taisyklės reprezentaciją
```

```
# "Produkcijos vardas: prerekvizitai, atskirti kableliais -> išvada"
sub _produkcijos_aprasas {
    my( $taisykle, $number ) = @_;
    return "R" . $number . " : "
        . join( ", ", @{ $taisykle->{'premises'} } )
        . " -> " . $taisykle->{'conclusion'};
}

# Funkcija, grąžinanti eilutę, su žingsnio numeriu, reikiamu atitraukimu ir tikslu
sub _sp_tikslas {
    my ($step, $tab, $tikslas) = @_;
    return sprintf "%3d. %s", $step, ' ' x 2 x $tab . "Tikslas $tikslas.";
}

# Funkcija, grąžinanti verifikavimo grafą pseudografiniu pavidalu
sub _verifikavimo_grafas {
    my $taisykliu_seka = shift;
    my @faktai = map { @$_ } @_;
    my $pradiniu_faktu_sk = @{ $_[0] };
    # Sukuriamas tuščių eilučių masyvas,
    # prie kurių bus iteratyviai priklijuojama pseudografika.
    my @lines = ('') x (@faktai +2);
    for my $i ($pradiniu_faktu_sk +2 .. @faktai +2){
        # Skaitliukas, kiek šioje iteracijoje reikės spausdinti pseudografikos
        my $jj = 0;
        # Priklijuojama pseudografinė aibė (užpildyto stačiakampio pavidalu)
        for my $j (1 .. @faktai +2){
            if( $j > (@faktai +2 - $i) >> 1 and $jj ++ < $i ){
                $lines[ $j-1 ] .= do {
                    if   ( $jj ==  1 ){ '┌' }
                    elsif( $jj == $i ){ '└' }
                    else {
                        '│' . $faktai[ $jj-2 ] . '│'
                    }
                };
            }
            else {
                $lines[ $j-1 ] .= ' ' x 3;
            }
        }

        last if $i == @faktai +2;
        # Priklijuojamos rodyklytės su taisyklėmis
        for my $j (1 .. @faktai +2){
            $lines[ $j-1 ] .= do {
                if( $j == (@faktai +2 +1) >> 1 ){
                    '-' . 'R' . $taisykliu_seka->[$i - $pradiniu_faktu_sk -2]. '→'
                }
                else {
                ' ' x (3 + length $taisykliu_seka->[$i - $pradiniu_faktu_sk -2]);
                }
            };
        }
    }
    return join "\n", "Verifikavimo grafas:", @lines
}
```

6. Literatūra

1. Straipsnis apie tiesioginį išvedimą Wikipedia.org *http://en.wikipedia.org/wiki/Backward_chaining*
2. V. Čyras. Intelektualios sistemos. *http://www.mif.vu.lt/~cyras/konspektas-intelektualios-sistemos.pdf*, p. 93-102. [2015-11-05]