VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Tiesioginis ir atbulinis išvedimas produkcijų sistemoje „Python“ programavimo kalba

Darbą atliko
Informatikos 4 kurso 4 grupės
studentas Redas Jatkauskas

Vilnius
2019

Turinys

1. Tiesioginio išvedimo algoritmas

1.1 Algoritmo žingsniai

1) Patikrinama, ar tarp faktų yra tikslas. Jei taip, grąžinama sėkmė.
2) Patikrinama, ar yra produkcijų, kurioms nepakelta flag1 arba flag2. Jei nėra, grąžinama nesėkmė
3) Perrenkamos produkcijos. Ieškoma tokios produkcijos, kuriai:
   a. Nepakelta flag1 arba flag2
   b. Nėra konsekvento. Jeigu yra, produkcijai pakeliama flag2.
   c. Visi kairėje pusėje pusėje esantys kintamieji yra tarp faktų
4) Radus produkciją, jos dešinėje pusėje esantis kintamasis pridedamas prie faktų. Produkcijai pakeliama flag1. Kartojama nuo 1 žingsnio.

1.2 Algoritmo pseudokodas

Įvestis:

- rules – produkcijų sąrašas
- facts – faktų sąrašas
- goal – tikslas

Išvestis:

- boolean – algoritmas gražina **True**, jei kelias iki tikslo egzistuoja. Kitu atveju – **False**
- road – sąrašas, kuriame kaupiamos tikslo radimo produkcijos

```
function forward_chaining(rules, facts, goal) returns boolean
        road := []
        repeat
                if goal in facts then
                        return True, road
                end if
                rule_found := False
                for all rule in rules
                        if not flag1(rule) and not flag2(rule) and is_statisfied(rule, facts) then
                                if right_side(rule) in facts then
                                        set_flag2(rule, True)
                                else
                                        rule_found := True
                                        road := road ∪ { right_side(rule) }
                                        facts := facts ∪ { name(rule) }
                                end if
                        end if
                end for
        until not rule_found
        return False
end function
```

1.3 Programos veikimo pavyzdžiai

1.3.1 Faktas konsekvente

Įvestis:

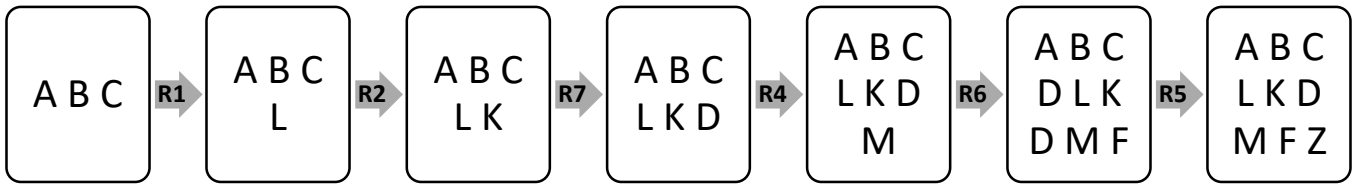
```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-26
# 1 testas
# 1) Taisyklės:
L A   # R1: A -> L
K L   # R2: L -> K
A D   # R3: D -> A
M D   # R4: D -> M
Z F B # R5: F, B -> Z
F C D # R6: C, D -> F
D A   # R7: A -> D

# 2) Faktai:
A B C

# 3) Tikslas:
Z
```

Išvestis:

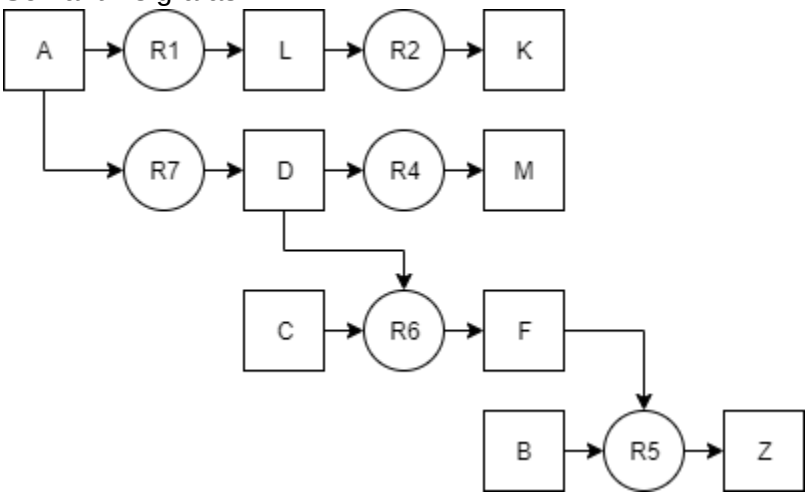
```
1 DALIS. Duomenys

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

  2) Faktai A, B, C.

  3) Tikslas Z.

2 DALIS. Vykdymas

  1 ITERACIJA
    R1:A->L taikome. Pakeliame flag1. Faktai A, B, C ir L.

  2 ITERACIJA
    R1:A->L praleidžiame, nes pakelta flag1.
    R2:L->K taikome. Pakeliame flag1. Faktai A, B, C ir L, K.

  3 ITERACIJA
    R1:A->L praleidžiame, nes pakelta flag1.
    R2:L->K praleidžiame, nes pakelta flag1.
    R3:D->M netaikome, nes trūksta D
    R4:D->M netaikome, nes trūksta D
    R5:F,B->Z netaikome, nes trūksta F
    R6:C,D->F netaikome, nes trūksta D
    R7:A->D taikome. Pakeliame flag1. Faktai A, B, C ir L, K, D.

  4 ITERACIJA
    R1:A->L praleidžiame, nes pakelta flag1.
    R2:L->K praleidžiame, nes pakelta flag1.
    R3:D->M taikome. Pakeliame flag1. Faktai A, B, C ir L, K, D, M.

  5 ITERACIJA
```

```
    R1:A->L praleidžiame, nes pakelta flag1.
    R2:L->K praleidžiame, nes pakelta flag1.
    R3:D->M praleidžiame, nes pakelta flag1.
    R4:D->M netaikome, nes konsekventas faktuose. Pakeliame flag2.
    R5:F,B->Z netaikome, nes trūksta F
    R6:C,D->F taikome. Pakeliame flag1. Faktai A, B, C ir L, K, D, M, F.

  6 ITERACIJA
    R1:A->L praleidžiame, nes pakelta flag1.
    R2:L->K praleidžiame, nes pakelta flag1.
    R3:D->M praleidžiame, nes pakelta flag1.
    R4:D->M praleidžiame, nes pakelta flag2.
    R5:F,B->Z taikome. Pakeliame flag1. Faktai A, B, C ir L, K, D, M, F, Z.

3 DALIS. Rezultatai

  1) Tikslas Z išvestas.
  2) Kelias: R1, R2, R7, R3, R6, R5.
```

Verifikavimo grafas:



Semantinis grafas:



1.3.2 Čyras vs. Negnevitsky; Čyras laimi

Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-26
# 2 testas
# 1) Taisyklės:
Z G # R1: G -> Z
G A # R2: A -> G
B A # R3: A -> B
C B # R4: B -> C
D C # R5: C -> D
Z D # R6: D -> Z

# 2) Faktai:
```

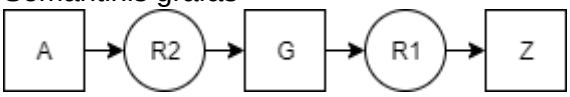
```
A

# 3) Tikslas:
Z
```

Išvestis:

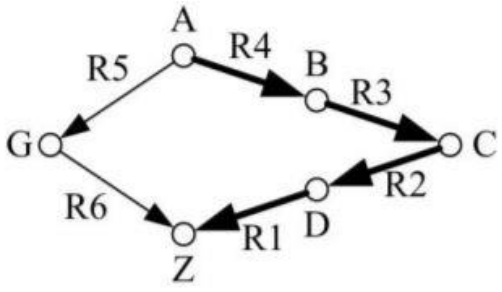
```
1 DALIS. Duomenys

  1) Taisyklės
    R1: G->Z
    R2: A->G
    R3: A->B
    R4: B->C
    R5: C->D
    R6: D->Z

  2) Faktai A.

  3) Tikslas Z.

2 DALIS. Vykdymas

  1 ITERACIJA
    R1:G->Z netaikome, nes trūksta G
    R2:A->G taikome. Pakeliame flag1. Faktai A ir G.

  2 ITERACIJA
    R1:G->Z taikome. Pakeliame flag1. Faktai A ir G, Z.

3 DALIS. Rezultatai

  1) Tikslas Z išvestas.
  2) Kelias: R2, R1.
```

Verifikavimo grafas:

Semantinis grafas



1.3.3 Čyras vs. Negnevitsky; Negnevitsky laimi



Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-26
# 3 testas
# 1) Taisyklės:
Z D # R1: D -> Z
```

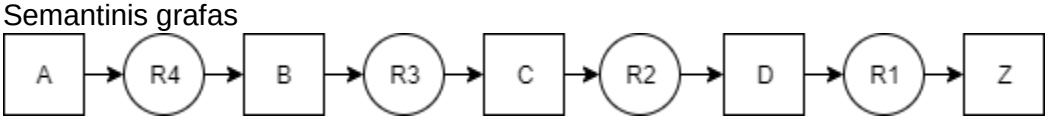
```
D C # R2: C -> D
C B # R3: B -> C
B A # R4: A -> B
G A # R5: A -> G
Z G # R6: G -> Z

# 2) Faktai:
A

# 3) Tikslas:
Z
```

Išvestis:

```
1 DALIS. Duomenys

  1) Taisyklės
    R1: D->Z
    R2: C->D
    R3: B->C
    R4: A->B
    R5: A->G
    R6: G->Z

  2) Faktai A.

  3) Tikslas Z.

2 DALIS. Vykdymas

  1 ITERACIJA
    R1:D->Z netaikome, nes trūksta D
    R2:C->D netaikome, nes trūksta C
    R3:B->C netaikome, nes trūksta B
    R4:A->B taikome. Pakeliame flag1. Faktai A ir B.

  2 ITERACIJA
    R1:D->Z netaikome, nes trūksta D
    R2:C->D netaikome, nes trūksta C
    R3:B->C taikome. Pakeliame flag1. Faktai A ir B, C.

  3 ITERACIJA
    R1:D->Z netaikome, nes trūksta D
    R2:C->D taikome. Pakeliame flag1. Faktai A ir B, C, D.

  4 ITERACIJA
    R1:D->Z taikome. Pakeliame flag1. Faktai A ir B, C, D, Z.

3 DALIS. Rezultatai

  1) Tikslas Z išvestas.
  2) Kelias: R4, R3, R2, R1.
```

Verifikavimo grafas:

Semantinis grafas



1.3.4 Tikslas tarp faktų

Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-26
# 4 testas
# 1) Taisyklės:
C A B # R1: A, B -> C
A B   # R2: B -> A

# 2) Faktai:
A B

# 3) Tikslas:
A
```

Išvestis:

```
1 DALIS. Duomenys

  1) Taisyklės
    R1: A,B->C
    R2: B->A

  2) Faktai A, B.

  3) Tikslas A.

2 DALIS. Vykdymas

3 DALIS. Rezultatai

  1) Tikslas A tarp faktų.
  2) Kelias tuščias.
```

Verifikavimo grafas:

Semantinis grafas:

1.3.5 Kelias neegzistuoja

Įvestis:

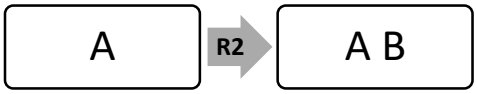
```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-26
# 5 testas
# 1) Taisyklės:
B A # R1: A -> B
Z C # R2: C -> Z

# 2) Faktai:
A
```

```
# 3) Tikslas:
Z
```

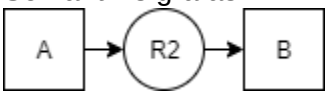
Išvestis:

```
1 DALIS. Duomenys

  1) Taisyklės
    R1: A->B
    R2: C->Z

  2) Faktai A.

  3) Tikslas Z.

2 DALIS. Vykdymas

  1 ITERACIJA
    R1:A->B taikome. Pakeliame flag1. Faktai A ir B.

  2 ITERACIJA
    R1:A->B praleidžiame, nes pakelta flag1.
    R2:C->Z netaikome, nes trūksta C

3 DALIS. Rezultatai

  1) Tikslo Z išvesti nepavyko.
```

Verifikavimo grafas:



Semantinis grafas:



1.3.6 Negnevitsky pavyzdys

Pavyzdys iš [Neg05] psl. 37.

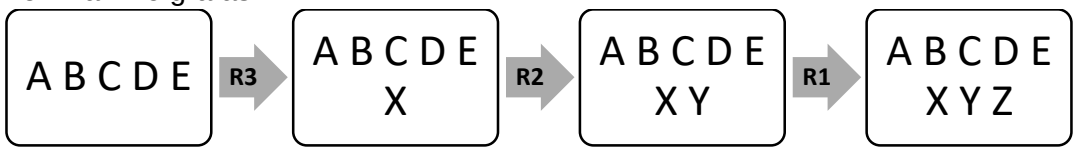
Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-26
# 6 testas
# 1) Taisyklės:
Z Y D   # R1: Y, D -> Z
Y X B E # R2: X, B, E -> Y
X A     # R3: A -> X
L C     # R4: C -> L
N L M   # R5: L, M -> N

# 2) Faktai:
A B C D E

# 3) Tikslas:
Z
```

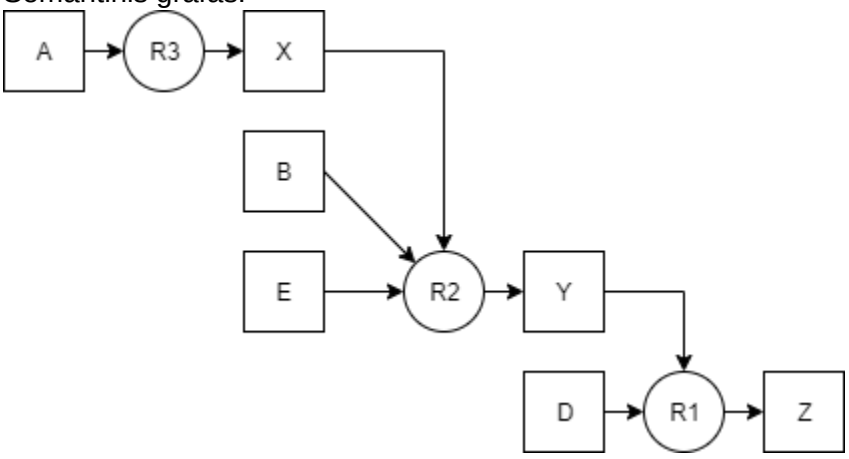
Išvestis:

```
1 DALIS. Duomenys

  1) Taisyklės
    R1: Y,D->Z
    R2: X,B,E->Y
    R3: A->X
    R4: C->L
    R5: L,M->N

  2) Faktai A, B, C, D, E.

  3) Tikslas Z.

2 DALIS. Vykdymas

  1 ITERACIJA
    R1:Y,D->Z netaikome, nes trūksta Y
    R2:X,B,E->Y netaikome, nes trūksta X
    R3:A->X taikome. Pakeliame flag1. Faktai A, B, C, D, E ir X.

  2 ITERACIJA
    R1:Y,D->Z netaikome, nes trūksta Y
    R2:X,B,E->Y taikome. Pakeliame flag1. Faktai A, B, C, D, E ir X, Y.

  3 ITERACIJA
    R1:Y,D->Z taikome. Pakeliame flag1. Faktai A, B, C, D, E ir X, Y, Z.

3 DALIS. Rezultatai

  1) Tikslas Z išvestas.
  2) Kelias: R3, R2, R1.
```

Verifikavimo grafas:



Semantinis grafas:



1.4 Programos kodas

```
class Rule:

    def __init__(self, left, right):
        self.left = left
        self.right = right
        self.flag1 = False
        self.flag2 = False

    def follows(self, facts):

        for fact in self.left:
            if fact not in facts:
                return fact
        return None

    def __str__(self):
        return ",".join(self.left) + "->" + self.right


class ForwardChaining:

    def __init__(self, file_name):
        self.iteration = 0
        self.output = ""
        self.output_file_name = None

        self.output += "1 DALIS. Duomenys\n"
        rules, facts, goal = self.read_data(file_name)
        self.print_data(rules, facts, goal)

        self.output += "2 DALIS. Vykdymas\n"
        result, road = self.forward_chaining(rules, facts, goal)

        self.output += "3 DALIS. Rezultatai\n"
        self.print_results(result, road, goal)

        self.write_output(file_name)
        print("Rezultatas išsaugotas faile: %s." % self.output_file_name)

    def forward_chaining(self, rules, facts, goal):
        ir = len(facts)
        iteration = 0
        road = []

        while goal not in facts:
            rule_applied = False
            iteration += 1
            self.output += "%i".rjust(4, " ") % iteration + " ITERACIJA\n"

            for rule in rules:
                self.output += "    R%i:%s " % ((rules.index(rule) + 1), str(rule))

                if rule.flag1:
                    self.output += "praleidžiame, nes pakelta flag1.\n"
                    continue

                if rule.flag2:
                    self.output += "praleidžiame, nes pakelta flag2.\n"
                    continue

                if rule.right in facts:
```

```
                    self.output += "netaikome, nes konsekventas faktuose. Pakeliame
flag2.\n"
                    rule.flag2 = True
                    continue

                missing = rule.follows(facts)

                if missing is None:
                    rule_applied = True
                    rule.flag1 = True
                    facts.append(rule.right)
                    road.append("R" + str(rules.index(rule) + 1))
                    self.output += "taikome. Pakeliame flag1. Faktai %s ir %s.\n" %(
                        ", ".join(facts[:ir]), ", ".join(facts[ir:]))
                    break
                else:
                    self.output += "naetaikome, nes trūksta %s\n" % missing
            self.output += "\n"

            if not rule_applied:
                return False, []

        return True, road

    def read_data(self, file_name):
        rules = []
        facts = []
        goal = None

        file = open(file_name, "r")
        read_state = 0

        for line in file:
            line = line.replace("\n", "")

            if line == "":
                read_state += 1
                continue
            if line[0] == '#':
                continue

            line = line.split(" ")

            if read_state == 0:
                right = line[0]
                left = line[1:]
                rules.append(Rule(left, right))

            if read_state == 1:
                facts = line

            if read_state == 2:
                goal = line[0]

            if read_state > 2:
                self.output += "Neteisingas duomenų failas!"
                return [], [], None

        return rules, facts, goal

    def print_data(self, rules, facts, goal):

        self.output += "  1) Taisyklės\n"
```

```
        for rule in rules:
            self.output += "    R%i: %s\n" % (rules.index(rule) + 1, str(rule))
        self.output += "\n  2) Faktai %s.\n" % ", ".join(facts)
        self.output += "\n  3) Tikslas %s\n\n" % goal

    def print_results(self, result, road, goal):

        if result:
            if len(road) == 0:
                self.output += "  1) Tikslas %s tarp faktų.\n" % goal
                self.output += "  2) Kelias tuščias.\n"
            else:
                self.output += "  1) Tikslas %s išvestas.\n" % goal
                self.output += "  2) Kelias: %s.\n" % ", ".join(road)
        else:
            self.output += "  1) Tikslo %s išvesti nepavyko.\n" % goal

    def write_output(self, file_name):
        self.output_file_name = "out/FC_OUTPUT_%s.txt" % file_name.replace("/", ".")
        file = open(self.output_file_name, "w", encoding='utf8')
        file.write(self.output)
```

2. Atbulinio išvedimo algoritmas

2.1 Algoritmo žingsniai

1) Patikrinama, ar esamas tikslas yra tarp ieškomų faktų. Jei taip, grąžinama sėkmė.
2) Patikrinama, ar esamas tikslas yra tarp ieškomų tikslų sąrašo. Jei taip, reiškia ciklas. Grąžinama nesėkmė
3) Patikrinama, ar esamas tikslas yra tarp rastų faktų. Jei taip, grąžinama sėkmė.
4) Perrenkamos produkcijos:
    a. Patikrinama, ar jos dešinė pusė sutampa su esamu tikslu. Jeigu ne, einama prie kitos produkcijos
    b. Jeigu sutampa, kiekvienam kintamajam kairėje produkcijos pusėje rekursyviai kviečiamas atbulinio išvedimo metodas.
    c. Jeigu su visais kintamaisiais gaunama sėkmė, produkcija pridedama prie sėkmės gavimo kelio. Grąžinama sėkmė.
5) Perrinkus visas produkcijas be sugrįžimo, grąžinama nesėkmė.

2.2 Algoritmo pseudokodas

Įvestis:

- Rules – produkcijų sąrašas
- target_facts – faktų sąrašas
- found_facts – sąrašas, kuriame kaupiami pereiti faktai
- current_goals – sąrašas, kuriame kaupiami ieškomi tikslai (naudojamas, kad išvengti ciklų)
- goal - tikslas

Išvestis:

- boolean – algoritmas gražina **True**, jei kelias iki tikslo egzistuoja. Kitu atveju – **False**
- road – sąrašas, kuriame kaupiamos tikslo radimo produkcijos

```
function backward_chaining(rules, target_facts, found_facts, current_goals, goal) returns
boolean
        if goal in target_facts
                return True # faktas (duotas)
        if goal in current_goals
                return False # ciklas
        If goal in found_facts
                return True # faktas (buvo gautas)
        for all rule in rules
                if rule.right = goal then
                        for all new_goal in rule.left
                                is_statisfied := backward_chaining(rules.remove(rule),
                                                                   target_facts,
                                                                   found_facts,
                                                                   current_goals.append(goal),
                                                                   new_goal)
                                if not is_statisfied
                                        break
                                if goal in found_facts
                                        return True # faktas (dabar gautas)
                        if is_statisfied
                                road.add(rule)
                                return True
        return False # Nera taisykliu
end function
```

2.3 Programos veikimo pavyzdžiai

2.3.1 Užmirštama šaka

Įvestis:

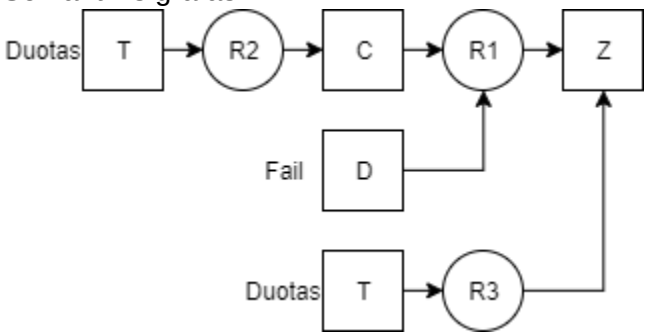
```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 1 testas
# 1) Taisyklės:
Z C D # R1: C, D -> Z
C T   # R2: T -> C
Z T   # R3: T -> Z

# 2) Faktai:
T

# 3) Tikslas:
Z
```

Išvestis:

```
1 DALIS. Duomenys
  1) Taisyklės
    R1: C,D->Z
    R2: T->C
    R3: T->Z

  2) Faktai
    T.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  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), nes faktai T. Grįžtame, sėkmė.
  4) -Tikslas C. Faktas (dabar gautas). Faktai T ir C. Grįžtame, sėkmė.
  5) -Tikslas D. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
  6) Tikslas Z. Randame R3:T->Z. Nauji tikslai T.
  7) -Tikslas T. Faktas (duotas), nes faktai T. Grįžtame, sėkmė.
  8) Tikslas Z. Faktas (dabar gautas). Faktai T ir C, Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R3.
```

Verifikavimo grafas:

Semantinis grafas:



2.3.2 Devynios produkcijos D, C

Įvestis:

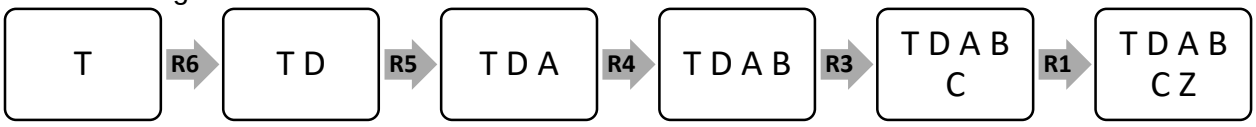
```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 2 testas
# 1) Taisyklės:
Z D C # R1: D, C -> Z
D C   # R2: C -> D
C B   # R3: B -> C
B A   # R4: A -> B
A D   # R5: D -> A
D T   # R6: T -> D
A G   # R7: G -> A
B H   # R8: H -> B
C J   # R9: J -> C

# 2) Faktai:
T

# 3) Tikslas:
Z
```

Išvestis:

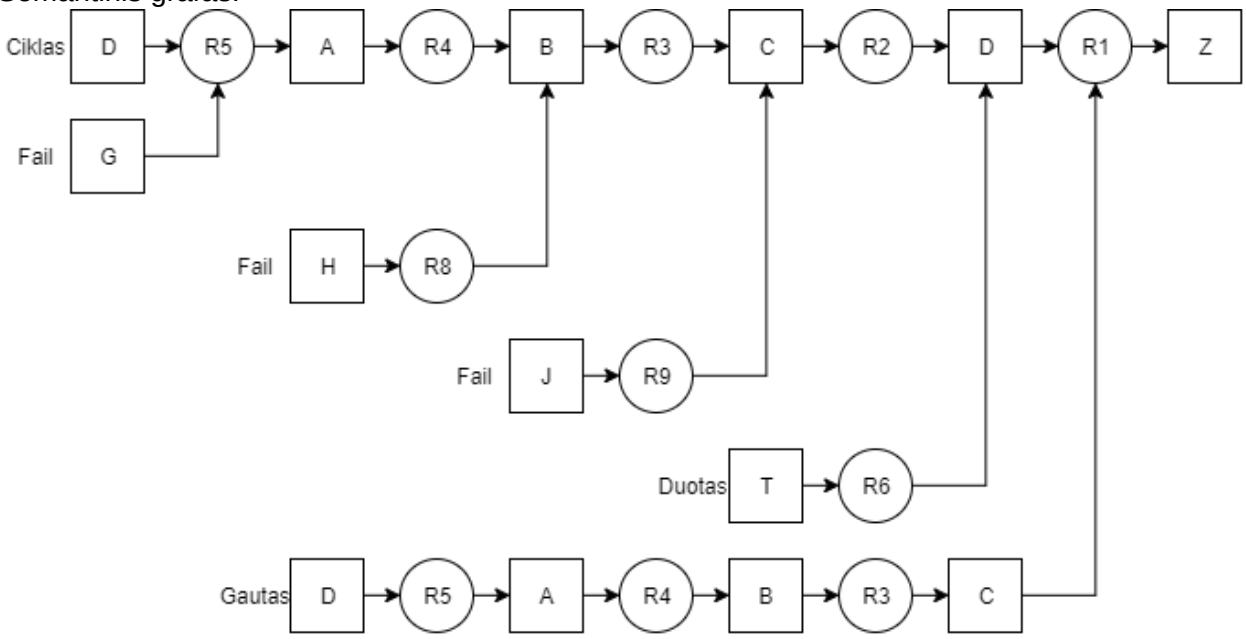
```
1 DALIS. Duomenys
  1) Taisyklės
    R1: D,C->Z
    R2: C->D
    R3: B->C
    R4: A->B
    R5: D->A
    R6: T->D
    R7: G->A
    R8: H->B
    R9: J->C

  2) Faktai
    T.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  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. Ciklas. Grįžtame, FAIL
  7) ----Tikslas A. Randame R7:G->A. Nauji tikslai G.
  8) -----Tikslas G. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
  9) ----Tikslas A. Nėra daugiau taisyklių jo išvedimui. Grįžtame, FAIL.
 10) ---Tikslas B. Randame R8:H->B. Nauji tikslai H.
 11) ----Tikslas H. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
 12) ---Tikslas B. Nėra daugiau taisyklių jo išvedimui. Grįžtame, FAIL.
 13) --Tikslas C. Randame R9:J->C. Nauji tikslai J.
 14) ---Tikslas J. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
 15) --Tikslas C. Nėra daugiau taisyklių jo išvedimui. Grįžtame, FAIL.
 16) -Tikslas D. Randame R6:T->D. Nauji tikslai T.
 17) --Tikslas T. Faktas (duotas), nes faktai T. Grįžtame, sėkmė.
 18) -Tikslas D. Faktas (dabar gautas). Faktai T ir D. Grįžtame, sėkmė.
 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 gautas), nes faktai T ir D. Grįžtame, sėkmė.
 23) ---Tikslas A. Faktas (dabar gautas). Faktai T ir D, A. Grįžtame, sėkmė.
 24) --Tikslas B. Faktas (dabar gautas). Faktai T ir D, A, B. Grįžtame, sėkmė.
 25) -Tikslas C. Faktas (dabar gautas). Faktai T ir D, A, B, C. Grįžtame, sėkmė.
 26) Tikslas Z. Faktas (dabar gautas). Faktai T ir D, A, B, C, Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R6, R5, R4, R3, R1.
```

Verifikavimo grafas:



Semantinis grafas:



2.3.3 Devynios produkcijos C, D

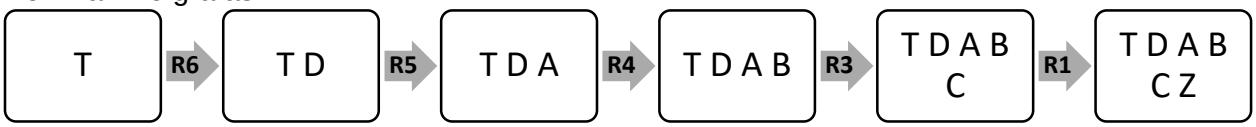
Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 3 testas
# 1) Taisyklės:
Z C D # R1: C, D -> Z
D C   # R2: C -> D
C B   # R3: B -> C
B A   # R4: A -> B
A D   # R5: D -> A
D T   # R6: T -> D
A G   # R7: G -> A
B H   # R8: H -> B
C J   # R9: J -> C

# 2) Faktai:
T

# 3) Tikslas:
Z
```

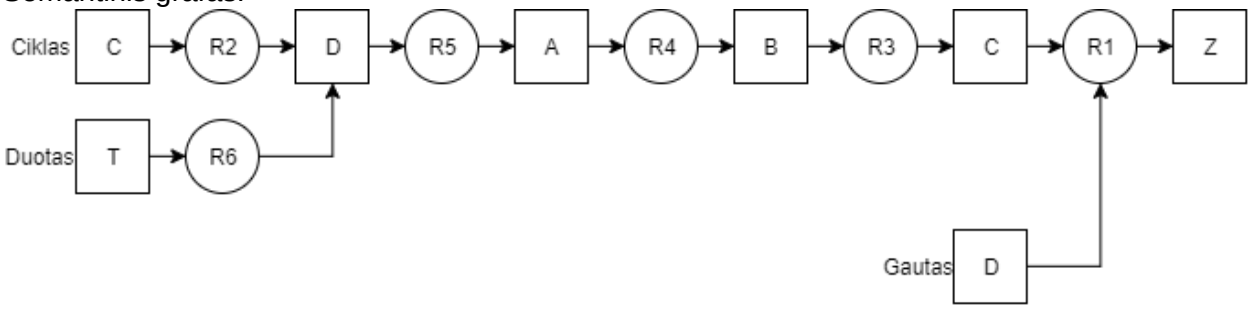
Išvestis:

```
1 DALIS. Duomenys
  1) 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

  2) Faktai
    T.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  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. Ciklas. Grįžtame, FAIL
  7) ----Tikslas D. Randame R6:T->D. Nauji tikslai T.
  8) -----Tikslas T. Faktas (duotas), nes faktai T. Grįžtame, sėkmė.
  9) ----Tikslas D. Faktas (dabar gautas). Faktai T ir D. Grįžtame, sėkmė.
 10) ---Tikslas A. Faktas (dabar gautas). Faktai T ir D, A. Grįžtame, sėkmė.
 11) --Tikslas B. Faktas (dabar gautas). Faktai T ir D, A, B. Grįžtame, sėkmė.
 12) -Tikslas C. Faktas (dabar gautas). Faktai T ir D, A, B, C. Grįžtame, sėkmė.
 13) -Tikslas D. Faktas (buvo gautas), nes faktai T ir D, A, B, C. Grįžtame, sėkmė.
 14) Tikslas Z. Faktas (dabar gautas). Faktai T ir D, A, B, C, Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R6, R5, R4, R3, R1.
```

Verifikavimo grafas:



Semantinis grafas:



2.3.4 Ciklas ir praleistas potikslis

Įvestis:

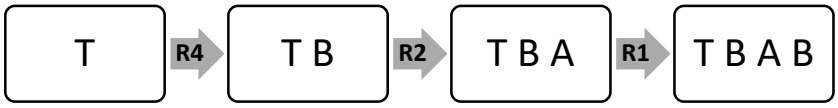
```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 4 testas
# 1) Taisyklės:
Z A   # R1: A -> Z
A B   # R2: B -> A
B A C # R3: A, C -> B
B T   # R4: T -> B
C T   # R5: T -> C

# 2) Faktai:
T

# 3) Tikslas:
Z
```

Išvestis:

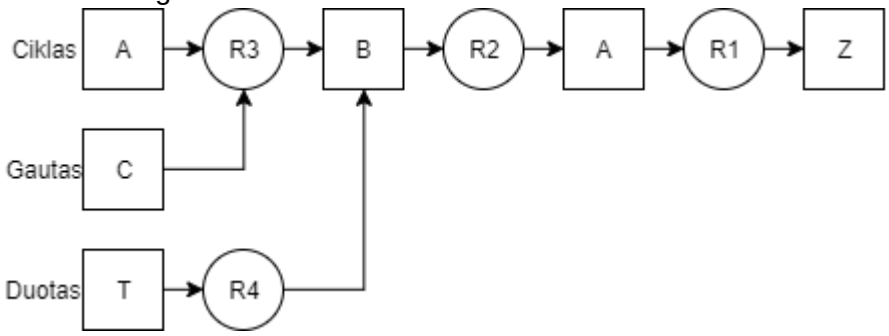
```
1 DALIS. Duomenys
  1) Taisyklės
    R1: A->Z
    R2: B->A
    R3: A,C->B
    R4: T->B
    R5: T->C

  2) Faktai
    T.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  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. Ciklas. Grįžtame, FAIL
  5) --Tikslas B. Randame R4:T->B. Nauji tikslai T.
  6) ---Tikslas T. Faktas (duotas), nes faktai T. Grįžtame, sėkmė.
  7) --Tikslas B. Faktas (dabar gautas). Faktai T ir B. Grįžtame, sėkmė.
  8) -Tikslas A. Faktas (dabar gautas). Faktai T ir B, A. Grįžtame, sėkmė.
```

```
  9) Tikslas Z. Faktas (dabar gautas). Faktai T ir B, A, Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R4, R2, R1.
```

Verifikavimo grafas:



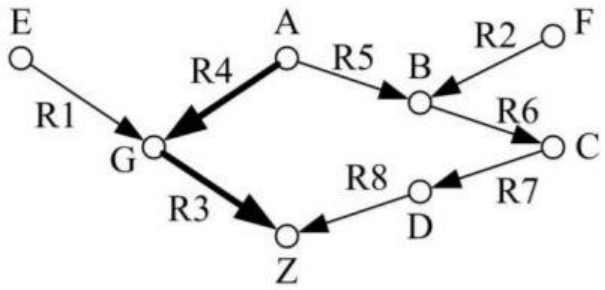
Semantinis grafas:



2.3.5 Grafas su trumpu keliu

Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 5 testas
# 1) Taisyklės:
G E # R1: E -> G
B F # R2: F -> B
Z G # R3: G -> Z
G A # R4: A -> G
B A # R5: A -> B
C B # R6: B -> C
D C # R7: C -> D
Z D # R8: D -> Z

# 2) Faktai:
A

# 3) Tikslas:
Z
```



Išvestis:

```
1 DALIS. Duomenys
  1) 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
```

```
  2) Faktai
    A.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  1) Tikslas Z. Randame R3:G->Z. Nauji tikslai G.
  2) -Tikslas G. Randame R1:E->G. Nauji tikslai E.
  3) --Tikslas E. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
  4) -Tikslas G. Randame R4:A->G. Nauji tikslai A.
  5) --Tikslas A. Faktas (duotas), nes faktai A. Grįžtame, sėkmė.
  6) -Tikslas G. Faktas (dabar gautas). Faktai A ir G. Grįžtame, sėkmė.
  7) Tikslas Z. Faktas (dabar gautas). Faktai A ir G, Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R4, R3.
```

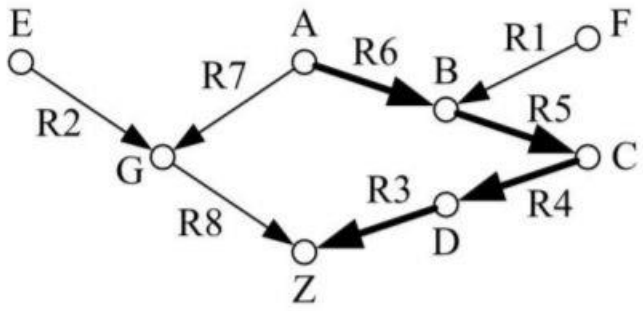
Verifikavimo grafas:

Semantinis grafas:

2.3.6 Grafas su ilgu keliu

Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 6 testas
# 1) Taisyklės:
B F # R1: F -> B
G E # R2: E -> G
Z D # R3: D -> Z
D C # R4: C -> D
C B # R5: B -> C
B A # R6: A -> B
G A # R7: A -> G
Z G # R8: G -> Z

# 2) Faktai:
A

# 3) Tikslas:
Z
```



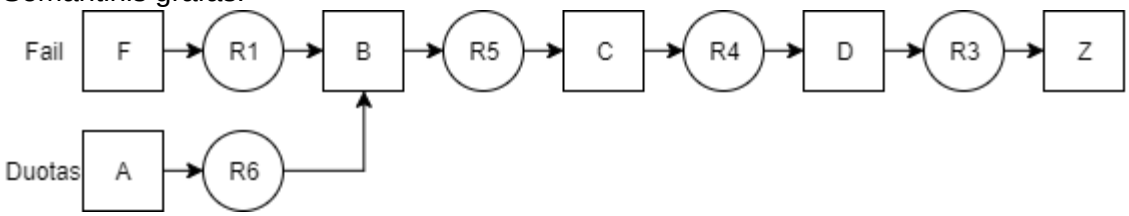
Išvestis:

```
1 DALIS. Duomenys
  1) Taisyklės
    R1: F->B
    R2: E->G
    R3: D->Z
    R4: C->D
    R5: B->C
    R6: A->B
```

```
    R7: A->G
    R8: G->Z

  2) Faktai
    A.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  1) Tikslas Z. Randame R3:D->Z. Nauji tikslai D.
  2) -Tikslas D. Randame R4:C->D. Nauji tikslai C.
  3) --Tikslas C. Randame R5:B->C. Nauji tikslai B.
  4) ---Tikslas B. Randame R1:F->B. Nauji tikslai F.
  5) ----Tikslas F. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
  6) ---Tikslas B. Randame R6:A->B. Nauji tikslai A.
  7) ----Tikslas A. Faktas (duotas), nes faktai A. Grįžtame, sėkmė.
  8) ---Tikslas B. Faktas (dabar gautas). Faktai A ir B. Grįžtame, sėkmė.
  9) --Tikslas C. Faktas (dabar gautas). Faktai A ir B, C. Grįžtame, sėkmė.
 10) -Tikslas D. Faktas (dabar gautas). Faktai A ir B, C, D. Grįžtame, sėkmė.
 11) Tikslas Z. Faktas (dabar gautas). Faktai A ir B, C, D, Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R6, R5, R4, R3.
```

Verifikavimo grafas:

Semantinis grafas:



2.3.7 Trys alternatyvos tikslui

Įvestis:

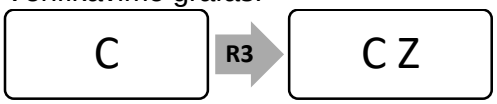
```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 7 testas
# 1) Taisyklės:
Z A # R1: A -> Z
Z B # R2: B -> Z
Z C # R3: C -> Z

# 2) Faktai:
C

# 3) Tikslas:
Z
```

Išvestis:

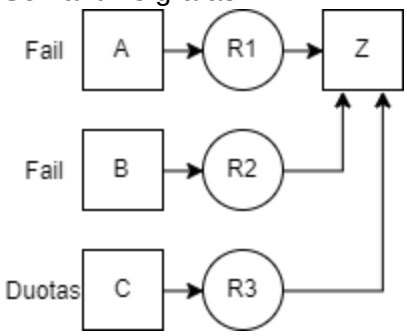
```
1 DALIS. Duomenys
  1) Taisyklės
    R1: A->Z
    R2: B->Z
```

```
    R3: C->Z

  2) Faktai
    C.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  1) Tikslas Z. Randame R1:A->Z. Nauji tikslai A.
  2) -Tikslas A. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
  3) Tikslas Z. Randame R2:B->Z. Nauji tikslai B.
  4) -Tikslas B. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
  5) Tikslas Z. Randame R3:C->Z. Nauji tikslai C.
  6) -Tikslas C. Faktas (duotas), nes faktai C. Grįžtame, sėkmė.
  7) Tikslas Z. Faktas (dabar gautas). Faktai C ir Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R3.
```

Verifikavimo grafas:



Semantinis grafas:



2.3.8 Nepasiekiamas tikslas

Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 8 testas
# 1) Taisyklės:
Z C D
Y C E

# 2) Faktai:
C D

# 3) Tikslas:
Y
```

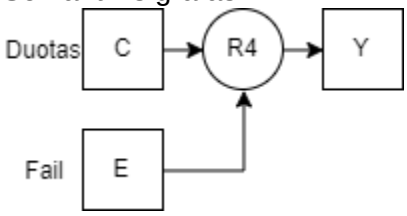
Išvestis:

```
1 DALIS. Duomenys
  1) Taisyklės
    R1: C,D->Z
    R2: C,E->Y

  2) Faktai
```

```
    C, D.

  3) Tikslas
    Y.

2 DALIS. Vykdymas
  1) Tikslas Y. Randame R2:C,E->Y. Nauji tikslai C, E.
  2) -Tikslas C. Faktas (duotas), nes faktai C, D. Grįžtame, sėkmė.
  3) -Tikslas E. Nėra taisyklių jo išvedimui. Grįžtame, FAIL.
  4) Tikslas Y. Nėra daugiau taisyklių jo išvedimui. Grįžtame, FAIL.

3 DALIS. Rezultatai
  1) Tikslo Y išvesti nepavyko.
```

Verifikavimo grafas:

Semantinis grafas:



2.3.9 Tikslas tarp faktų – tuščias kelias

Įvestis:

```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 9 testas
# 1) Taisyklės:
Q A # R1: A -> Q

# 2) Faktai:
Z

# 3) Tikslas:
Z
```

Išvestis:

```
1 DALIS. Duomenys
  1) Taisyklės
    R1: A->Q

  2) Faktai
    Z.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  1) Tikslas Z. Faktas (duotas), nes faktai Z. Grįžtame, sėkmė.

3 DALIS. Rezultatai
  1) Tikslas Z tarp faktų.
  2) Tuščias kelias.
```

Verifikavimo grafas:

Semantinis grafas:

2.3.10 Negnevitsky pavyzdys

Pavyzdys iš [Neg05] psl. 39.

Įvestis:

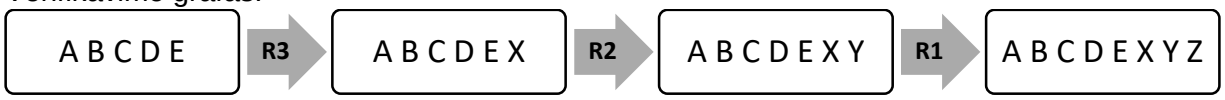
```
# Studentas Redas Jatkauskas, Informatika, 4 kursas, 4 grupė, 2019-11-30
# 10 testas
# 1) Taisyklės:
Z Y D   # R1: Y, D -> Z
Y X B E # R2: X, B, E -> Y
X A     # R3: A -> X
L C     # R4: C -> L
N L M   # R5: L, M -> N

# 2) Faktai:
A B C D E

# 3) Tikslas:
Z
```

Išvestis:

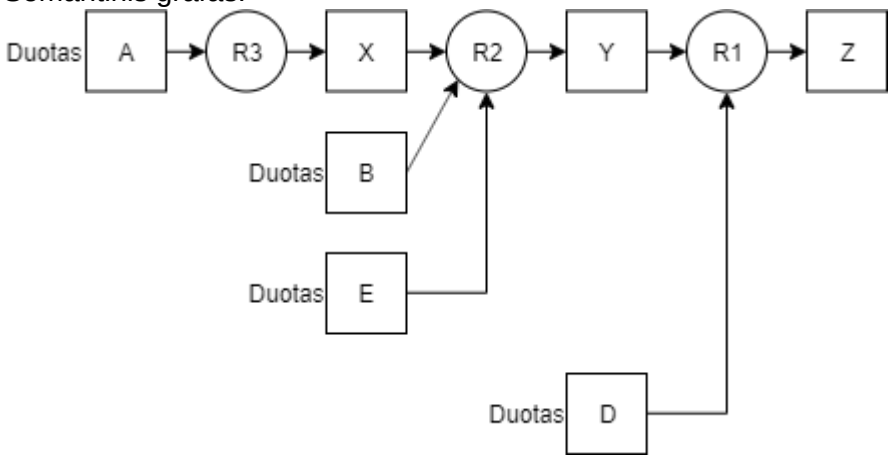
```
1 DALIS. Duomenys
  1) Taisyklės
    R1: Y,D->Z
    R2: X,B,E->Y
    R3: A->X
    R4: C->L
    R5: L,M->N

  2) Faktai
    A, B, C, D, E.

  3) Tikslas
    Z.

2 DALIS. Vykdymas
  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), nes faktai A, B, C, D, E. Griztame, sekme.
  5) --Tikslas X. Faktas (dabar gautas). Faktai A, B, C, D, E ir X. Griztame, sekme.
  6) --Tikslas B. Faktas (duotas), nes faktai A, B, C, D, E. Griztame, sekme.
  7) --Tikslas E. Faktas (duotas), nes faktai A, B, C, D, E. Griztame, sekme.
  8) -Tikslas Y. Faktas (dabar gautas). Faktai A, B, C, D, E ir X, Y. Griztame,
sekme.
  9) -Tikslas D. Faktas (duotas), nes faktai A, B, C, D, E. Griztame, sekme.
 10) Tikslas Z. Faktas (dabar gautas). Faktai A, B, C, D, E ir X, Y, Z. Griztame,
sekme.
```

```
3 DALIS. Rezultatai
  1) Tikslas Z išvestas.
  2) Kelias: R3, R2, R1.
```

Verifikavimo grafas:



Semantinis grafas:



2.4 Programos kodas

```
class Rule:
    def __init__(self, left, right):
        self.left = left
        self.right = right
        self.flag1 = False
        self.flag2 = False

    def follows(self, facts):
        for fact in self.left:
            if fact not in facts:
                return fact
        return None

    def __str__(self):
        return "%s->%s" % (",".join(self.left), self.right)


class BackwardChaining:
    def __init__(self, file_name):
        self.output = ""
        self.output_file_name = None
        self.iteration = 0
        self.current_goals = []
        self.found_facts = []
        self.road = []

        self.output += "1 DALIS. Duomenys\n"
        self.rules, self.target_facts, self.goal = self.read_data(file_name)
        self.print_data(self.rules, self.target_facts, self.goal)

        self.output += "2 DALIS. Vykdymas\n"
        result = self.do_backward_chaining(self.goal)
```

```
        self.output += "\n3 DALIS. Rezultatai\n"
        self.print_result(result)

        self.write_output(file_name)

    def do_backward_chaining(self, goal, indent=""):

        if goal in self.target_facts:
            self.print_step(goal, indent,
                            "Faktas (duotas), nes faktai %s. Griztame, sekme." % ",
".join(self.target_facts))
            return True

        if goal in self.current_goals:
            self.print_step(goal, indent, "Ciklas. Grįžtame, FAIL")
            return False

        if goal in self.found_facts:
            self.print_step(goal, indent, "Faktas (buvo gautas), nes faktai %s ir
%s. Griztame, sekme." % (
                ", ".join(self.target_facts), ", ".join(self.found_facts)))
            return True

        results_count = len(self.road)

        for rule in self.rules:
            if rule.right == goal:

                is_satisfied = False
                self.print_step(goal, indent, "Randame %s. Nauji tikslai %s." % (
                    "R" + str(self.rules.index(rule) + 1) + ":" + str(rule), ",
".join(rule.left)))

                for new_goal in rule.left:
                    self.current_goals.append(goal)
                    is_satisfied = self.do_backward_chaining(new_goal, indent + "-")
                    self.current_goals.pop()

                    if self.goal in self.found_facts:
                        return True

                if is_satisfied:
                    self.road.append("R" + str(self.rules.index(rule) + 1))
                    self.found_facts.append(rule.right)
                    self.print_step(goal, indent, "Faktas (dabar gautas). Faktai %s
ir %s. Griztame, sekme." % (
                        ", ".join(self.target_facts), ", ".join(self.found_facts)))
                    return True

            while len(self.road) > results_count:
                self.road.pop()

        self.print_step(goal, indent, "Nera taisykliu jo isvedimui. Griztame,
FAIL.")
        return False

    def print_step(self, goal, indent, msg):
        self.iteration += 1
        self.output += str(self.iteration).rjust(3, " ") + ") %sTikslas %s. " %
(indent, goal) + msg + "\n"

    def read_data(self, file_name):
```

```
        rules = []
        facts = []
        goal = None

        file = open(file_name, "r")
        read_state = 0

        for line in file:
            line = line.replace("\n", "")
            if line == "":
                read_state += 1
                continue
            line = line.split(" ")

            if line[0] == '#':
                continue

            if read_state == 0:
                right = line[0]
                left = line[1:]
                rules.append(Rule(left, right))

            if read_state == 1:
                facts = line

            if read_state == 2:
                goal = line[0]

        return rules, facts, goal

    def print_data(self, rules, facts, goal):
        self.output += "  1) Taisyklės\n"
        for rule in rules:
            self.output += "    R%i: %s\n" % (rules.index(rule) + 1, str(rule))
        self.output += "\n  2) Faktai\n    %s.\n\n" % ", ".join(facts)
        self.output += "  3) Tikslas\n    %s.\n\n" % goal

    def print_result(self, result):
        if result is not False:

            if len(self.road) == 0:
                self.output += "  1) Tikslas %s tarp faktų. Tuščias kelias.\n" %
self.goal
                self.output += "  2) Tuščias kelias.\n"
            else:
                self.output += "  1) Tikslas %s išvestas.\n" % self.goal
                self.output += "  2) Kelias: %s.\n" % ", ".join(self.road)
        else:
            self.output += "  1) Tikslo %s išvesti nepavyko.\n" % self.goal

    def write_output(self, file_name):
        self.output_file_name = "out/BC_OUTPUT_%s.txt" % file_name.replace("/", ".")
        file = open(self.output_file_name, "w", encoding='utf8')
        file.write(self.output)
```

3. Literatūros sąrašas

[Čyr19] V. Čyras. *Intelektualios sistemos.* http://klevas.mif.vu.lt/~cyras/AI/konspektas-intelektualios-sistemos.pdf 20 MB, 2019

[Neg05] M. Negnevitsky. *Artificial Intelligence. A Guide to Intelligent Systems.* Pearson Education Limited, Harlow, 2005, psl. 37-39.