VILNIAUS UNIVERSITETAS
MATEMATIKOS INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Tiesioginio ir atbulinio išvedimo laboratorinis darbas Java programavimo kalba

Laboratorinį darbą atliko:
Laimonas Beniušis
4 kursas, Kompiuterių Mokslas 1 grupė

Vilnius 2017

Turinys

Įvadas

Šiame dokumente yra pateikiama tiesioginio išvedimo „Forward chaining“algoritmo realizacija. Užduoties tikslas yra su pradine globalia duomenų baze rasti produkcijų seką, su kuria bus pasiekiamas tikslas (terminalinė būsena). Pradiniai uždavinio duomenys yra:

- Produkcijų (taisyklių) sąrašas
- Faktų sąrašas
- Tikslas

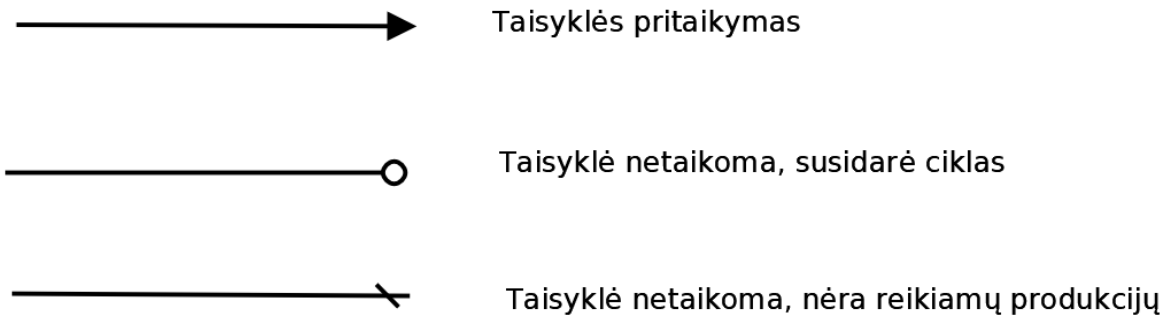
Programa parašyta naudojant Java programavimo kalbą. Dokumente pateikiami programos įvestis, išvestis, veikimo principai, testų pavyzdžiai, semantiniai grafai.

Produkciją galima pritaikyti jeigu jos dešiniosios pusės duomenys sutampa su turimu tikslu. Terminalinė būsena pasiekiama tada, kai tikslas yra išvedamas iš reikiamų faktų arba, panaudojus visas produkcijas tikslas nebuvo pasiektas. Produkcijos yra tikrinamos paeiliui.

Atbulinis išvedimas naudoja rekursiją ir stato darbinės atminties steką. Iškviesdamas naują funkcijos „rėmą“, turi atstatyti prieš tai buvusią būseną, neradus produkcijų kelio link tikslo.

1 Semantinių grafų aprašymas

Semantiniuose grafuose produkcijos (taisyklės) yra skaitomos iš viršaus į apačią ir iš kairės į dešinę.



2 Tiesioginis išvedimas

2.1 Pseodokodas

while (GDB būsena pakito) AND (tikslo nėra GDB) {
  for each rule {
    if (*anticedentas* sutampa su GBD esančiomis prielaidomis) AND (konsekventas pakeistų GDB) {
        įdėti pasirinktos taisyklės konsekventą į GDB
    }
}

2.2 Java implementacija

goal – tikslas (String)
db – globali duomenų bazė (duoti faktai) (HashSet<String>)
currentFacts – globalios duomenų bazės papildinys (faktai gauti algoritmo metu) (HashSet<String>)
makeCol – funkcija, apjungianti kelis „konteinerius“ į vieną HashSet

```
public boolean execute() {
    if (db.contains(goal)) return true;
    int i = 1;
    while (true) {
        int sizeBefore = makeCol(db, currentFacts).size();
        trace("\n" + i++ + " ITERATION");
        for (Rule rule : rules) {
            if (rule.flag1) {
                trace(rule + " skip, flag1");
            } else if (rule.flag2) {
                trace(rule + " skip, flag2");
            } else {
                boolean satisfied = true;
                HashSet<String> newDB = makeCol(db, currentFacts);
                for (String f : rule.left) {
                    if (!newDB.contains(f)) {
                        trace(rule + " skip, missing " + f);
                        satisfied = false;
                        break;
                    }
                }
                if (satisfied) {
                    if (newDB.contains(rule.right)) {
                        rule.flag2 = true;
                        trace(rule + " skip, consequent in DB");
                    } else {
                        results.add(rule.name);
                        currentFacts.add(rule.right);
                        newDB = makeCol(db, currentFacts);
                        rule.flag1 = true;
                        trace(rule + " apply, DB:" + newDB);
                        if (rule.right.equals(this.goal)) {
                            trace("Goal found, terminating");
                            return true;
                        }}}}}
        if (sizeBefore == makeCol(db, currentFacts).size()) {
            trace("No new facts. Terminating");
            return false;
       }}}
```

2.3 Testai

2.3.1 Faktas konsekvente

Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//1 testas. Faktas konsekvente.
//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

```
PART 1. Data
        1) Rules:
                R1: [A] -> L
                R2: [L] -> K
                R3: [D] -> A
                R4: [D] -> M
                R5: [F, B] -> Z
                R6: [C, D] -> F
                R7: [A] -> D

        2) Given facts:
                A
                B
                C

        3) Goal:
                Z

PART 2. Execution
```

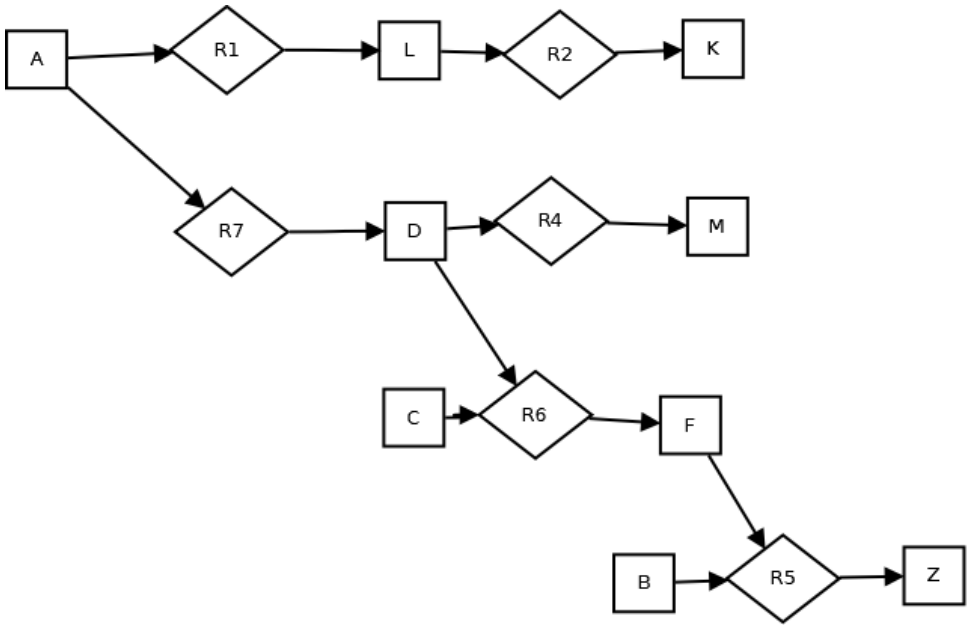
1 ITERATION
R1: [A] -> L apply, DB:[A, B, C, L]
R2: [L] -> K apply, DB:[A, B, C, K, L]
R3: [D] -> A skip, missing D
R4: [D] -> M skip, missing D
R5: [F, B] -> Z skip, missing F
R6: [C, D] -> F skip, missing D
R7: [A] -> D apply, DB:[A, B, C, D, K, L]

2 ITERATION
R1: [A] -> L skip, flag1
R2: [L] -> K skip, flag1
R3: [D] -> A skip, consequent in DB
R4: [D] -> M apply, DB:[A, B, C, D, K, L, M]
R5: [F, B] -> Z skip, missing F
R6: [C, D] -> F apply, DB:[A, B, C, D, F, K, L, M]
R7: [A] -> D skip, flag1

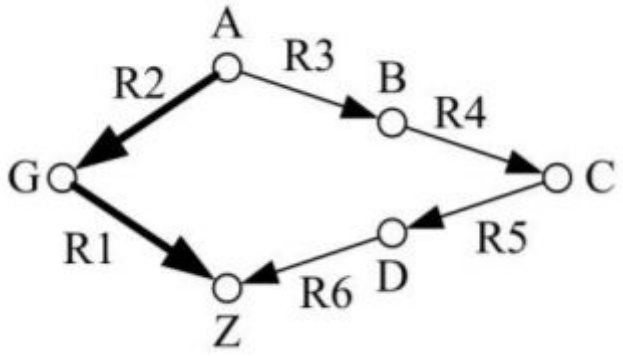
3 ITERATION
R1: [A] -> L skip, flag1
R2: [L] -> K skip, flag1
R3: [D] -> A skip, flag2
R4: [D] -> M skip, flag1
R5: [F, B] -> Z apply, DB:[A, B, C, D, F, Z, K, L, M]
Goal found, terminating

PART 3. Results
1) Z was deduced
2) Path: [R1, R2, R7, R4, R6, R5]

Semantinis grafas



2.3.2 Čyras vs Negnevitsky; Čyras laimi



Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//2 testas. Čyras vs Negnevitsky; Čyras laimi
//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

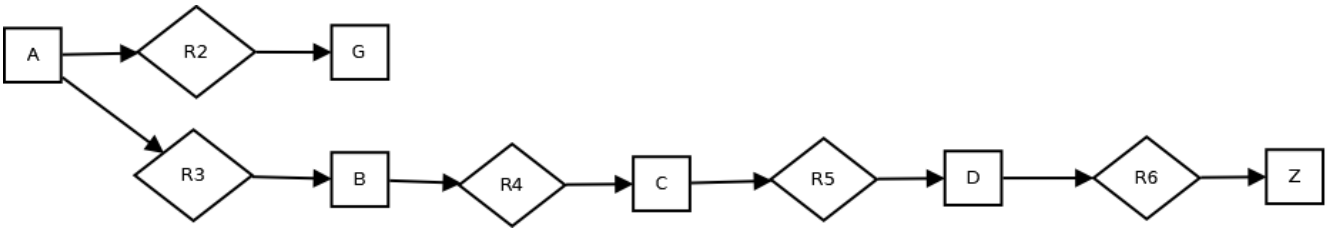
```
PART 1. Data
        1) Rules:
                R1: [G] -> Z
                R2: [A] -> G
                R3: [A] -> B
                R4: [B] -> C
                R5: [C] -> D
                R6: [D] -> Z

        2) Given facts:
                A

        3) Goal:
                Z
```

PART 2. Execution

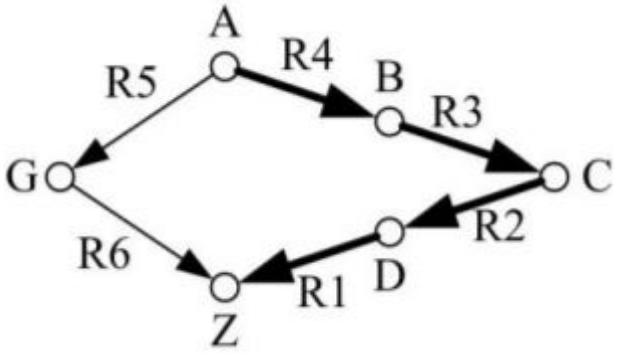
1 ITERATION
R1: [G] -> Z skip, missing G
R2: [A] -> G apply, DB:[A, G]
R3: [A] -> B apply, DB:[A, B, G]
R4: [B] -> C apply, DB:[A, B, C, G]
R5: [C] -> D apply, DB:[A, B, C, D, G]
R6: [D] -> Z apply, DB:[A, B, C, D, G, Z]
Goal found, terminating

PART 3. Results
1) Z was deduced
2) Path: [R2, R3, R4, R5, R6]

Semantinis grafas



2.3.3 Čyras vs Negnevitsky; Negnevitsky laimi



Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//3 testas. Čyras vs Negnevitsky; Negnevitsky laimi
//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

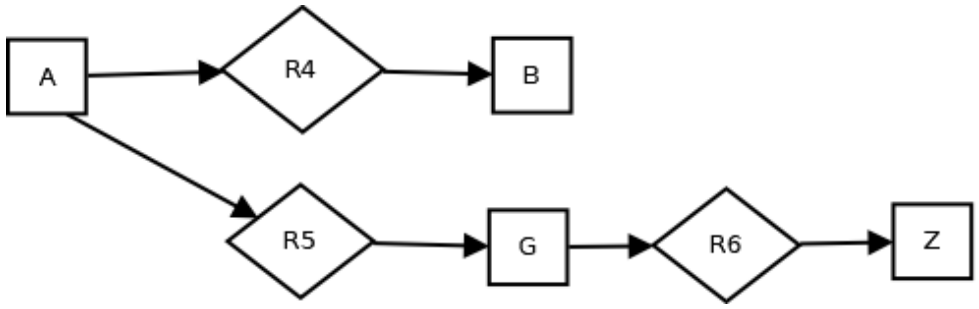
```
PART 1. Data
        1) Rules:
                R1: [D] -> Z
                R2: [C] -> D
                R3: [B] -> C
                R4: [A] -> B
                R5: [A] -> G
                R6: [G] -> Z

        2) Given facts:
                A

        3) Goal:
                Z
```

PART 2. Execution

1 ITERATION
R1: [D] -> Z skip, missing D
R2: [C] -> D skip, missing C
R3: [B] -> C skip, missing B
R4: [A] -> B apply, DB:[A, B]
R5: [A] -> G apply, DB:[A, B, G]
R6: [G] -> Z apply, DB:[A, B, G, Z]
Goal found, terminating

PART 3. Results
1) Z was deduced
2) Path: [R4, R5, R6]

Semantinis grafas



2.3.4 Tikslas tarp faktų

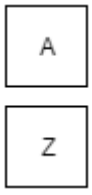
Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//4 testas. Tikslas tarp faktų
//1) Taisyklės:
Z A // R1: A -> Z

//2) Faktai:
A Z

//3) Tikslas:
Z
```

Išvestis

```
PART 1. Data
	1) Rules:
		R1: [A] -> Z

	2) Given facts:
		A
		Z

	3) Goal:
		Z

PART 2. Execution


PART 3. Results
	1) Z was given
```

Semantinis grafas



2.3.5 Kelias neegzistuoja

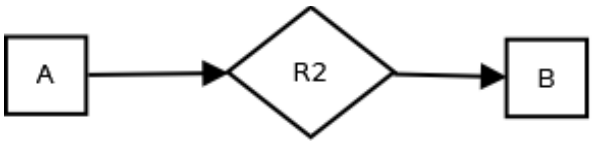
Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//5 testas. Kelias neegzistuoja
//1) Taisyklės:
B A // R1: A -> B
Z C // R2: C -> Z

//2) Faktai:
A

//3) Tikslas:
Z
```

Išvestis

```
PART 1. Data
        1) Rules:
                R1: [A] -> B
                R2: [C] -> Z

        2) Given facts:
                A

        3) Goal:
                Z

PART 2. Execution


1 ITERATION
R1: [A] -> B apply, DB:[A, B]
R2: [C] -> Z skip, missing C

2 ITERATION
R1: [A] -> B skip, flag1
R2: [C] -> Z skip, missing C
No new facts. Terminating

PART 3. Results
        1) Z was not deduced
```

Semantinis grafas



2.3.6 Negnevitsky pavyzdys (5 produkcijos, p.37)

Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//6 testas. Negnevitsky pavyzdys (5 produkcijos, p.37)
//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) Tiklas:
Z
```

Išvestis

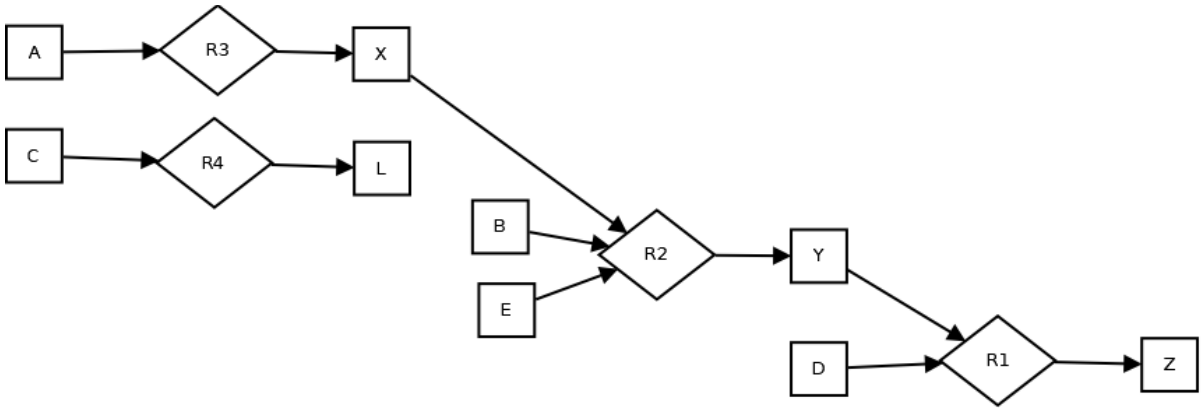
```
PART 1. Data
        1) Rules:
                R1: [Y, D] -> Z
                R2: [X, B, E] -> Y
                R3: [A] -> X
                R4: [C] -> L
                R5: [L, M] -> N

        2) Given facts:
                A
                B
                C
                D
                E

        3) Goal:
                Z

PART 2. Execution


1 ITERATION
R1: [Y, D] -> Z skip, missing Y
R2: [X, B, E] -> Y skip, missing X
R3: [A] -> X apply, DB:[A, B, C, D, E, X]
R4: [C] -> L apply, DB:[A, B, C, D, E, X, L]
R5: [L, M] -> N skip, missing M
```

2 ITERATION
R1: [Y, D] -> Z skip, missing Y
R2: [X, B, E] -> Y apply, DB:[A, B, C, D, E, X, Y, L]
R3: [A] -> X skip, flag1
R4: [C] -> L skip, flag1
R5: [L, M] -> N skip, missing M

3 ITERATION
R1: [Y, D] -> Z apply, DB:[A, B, C, D, E, X, Y, Z, L]
Goal found, terminating

PART 3. Results
1) Z was deduced
2) Path: [R3, R4, R2, R1]

Semantinis grafas



3 Atbulinis išvedimas

3.1 Pseodokodas

BackwardChain(G)
if(G yra darbinėje atmintyje) return true
if(nėra tokios taisyklės su konsekventu, kuris sutaptu su G) return false
for each rule {
	if(taisyklės konsekventas sutampa su G){
		if (visos pasirinktos taisykles prielados p BackwardChain(p) = true){
			return true
		}
	}
}
return false

3.2 Java implementacija

g – tikslas (String)
db – globali duomenų bazė (duoti faktai) (HashSet<String>)
currentFacts – globalios duomenų bazės papildinys (faktai gauti algoritmo metu) (HashSet<String>)
makeCol – funkcija, apjungianti kelis „konteinerius“ į vieną HashSet
correntGoals – tikslų sąrašas, saugantis nuo ciklų susidarymo, pradinis sąrašas yra tuščias
recFrame – rekursijos skaitliukas

```
private boolean bc(String g, LinkedList<String> currentGoals) {
    if (db.contains(g)) {
        return true;
    }
    boolean relevantRule = false;
    Object cloneFacts = currentFacts.clone();
    Object cloneResults = results.clone();
    for (Rule rule : rules) {
        if (rule.right.equals(g)) {
            relevantRule = true;
            boolean found = true;
            trace(format(g, "Find " + rule + ". New goals:" + rule.left));
            for (String f : rule.left) {
                if (db.contains(f)) {
                    trace(format(f, "Fact was given, DB:" + db));
                } else {
                    if (currentFacts.contains(f)) {
                        HashSet<String> newDB = makeCol(db, currentFacts);
                        trace(format(f, "Fact was deduced, DB:" + newDB));
                    } else {
                        if (currentGoals.contains(f)) {
                            trace(format(f, "Cycle detected, Backtracking. FAIL."));
                            found = false;
                            break;
                        }
                        currentGoals.addLast(f);
                        this.recFrame++;
                        found = bc(f, currentGoals);
                        this.recFrame--;
                        currentGoals.pollLast();
                        if (!found) {
                            break;
                        }
```

```
                        if (currentFacts.contains(this.goal)) {
                            return true;
                        }
                    }
                }
            }
            if (found) {
                results.add(rule.name);
                currentFacts.add(rule.right);
                HashSet<String> newDB = makeCol(db, currentFacts);
                trace(format(g, "New Fact added, DB:" + newDB + ", Backtracking. OK"));
                return true;
            }
        }
        currentFacts = (HashSet<String>) cloneFacts;
        results = (LinkedList<String>) cloneResults;
    }
    if (relevantRule) {
        trace(format(g, "No more rules to deduce it. Backtracking. FAIL"));
    } else {
        trace(format(g, "No rules to deduce it. Backtracking. FAIL"));
    }
    return false;
}
```

3.3 Testai

3.3.1 Užmirštama šaka

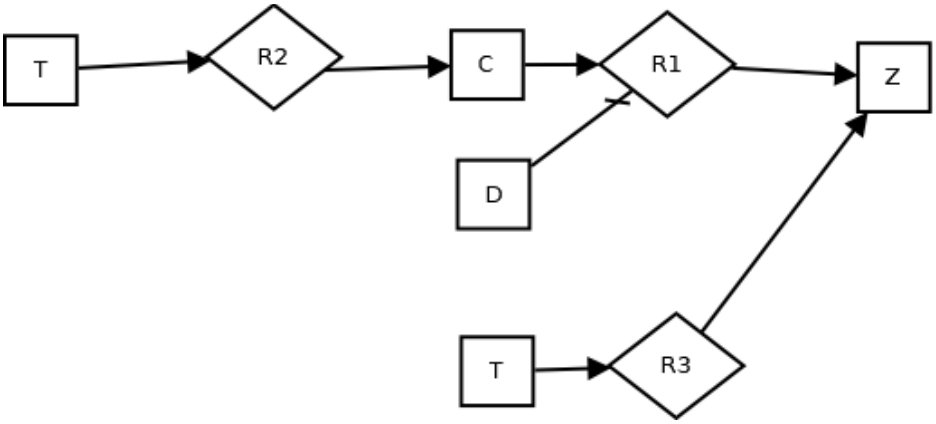
Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//1 testas. Užmirštama šaka.
//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

```
PART 1. Data
        1) Rules:
                R1: [C, D] -> Z
                R2: [T] -> C
                R3: [T] -> Z

        2) Given facts:
                T

        3) Goal:
                Z

PART 2. Execution

   1) .. Goal Z. Find R1: [C, D] -> Z. New goals:[C, D]
   2) .... Goal C. Find R2: [T] -> C. New goals:[T]
   3) .... Goal T. Fact was given, DB:[T]
   4) .... Goal C. New Fact added, DB:[C, T], Backtracking. OK
   5) .... Goal D. No rules to deduce it. Backtracking. FAIL
   6) .. Goal Z. Find R3: [T] -> Z. New goals:[T]
   7) .. Goal T. Fact was given, DB:[T]
   8) .. Goal Z. New Fact added, DB:[T, Z], Backtracking. OK

PART 3. Results
        1) Z was deduced
        2) Path: [R3]
```

Semantinis grafas



3.3.2 Devynios produkcijos D, C

Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//2 testas. Devynios produkcijos D, C.
//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

```
PART 1. Data
        1) Rules:
                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) Given facts:
T

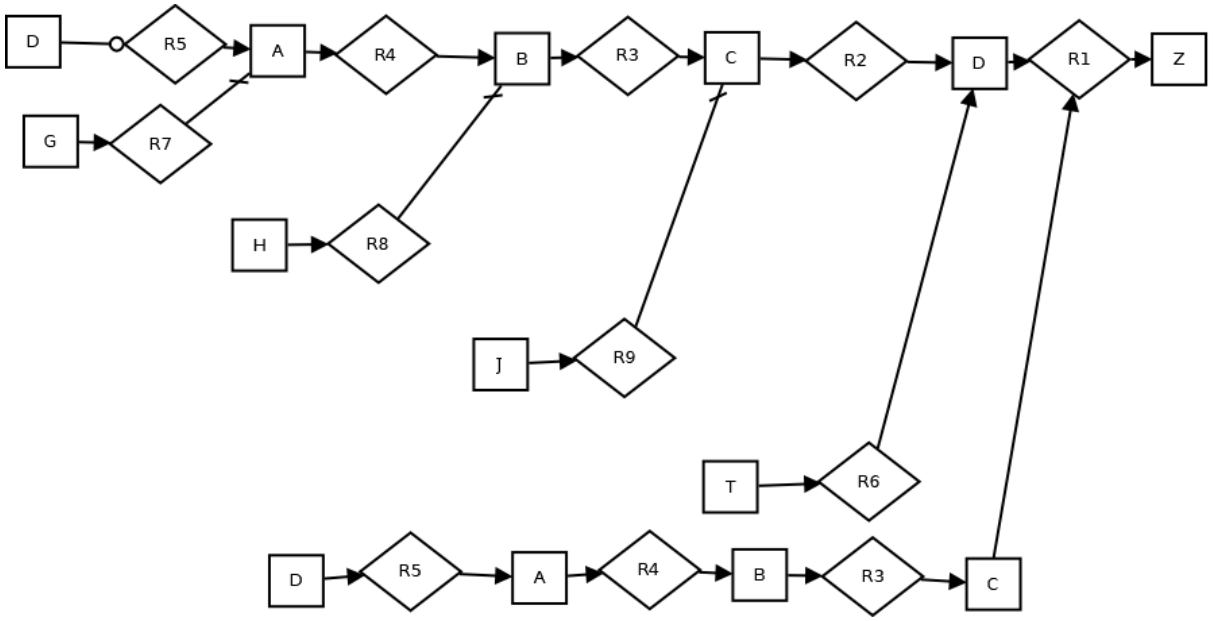
3) Goal:
Z

PART 2. Execution

1) .. Goal Z. Find R1: [D, C] -> Z. New goals:[D, C]
2) .... Goal D. Find R2: [C] -> D. New goals:[C]
3) ..... Goal C. Find R3: [B] -> C. New goals:[B]
4) ..... Goal B. Find R4: [A] -> B. New goals:[A]
5) ..... Goal A. Find R5: [D] -> A. New goals:[D]
6) ..... Goal D. Cycle detected, Backtracking. FAIL.
7) ..... Goal A. Find R7: [G] -> A. New goals:[G]
8) ..... Goal G. No rules to deduce it. Backtracking. FAIL
9) ..... Goal A. No more rules to deduce it. Backtracking. FAIL
10) ..... Goal B. Find R8: [H] -> B. New goals:[H]
11) ..... Goal H. No rules to deduce it. Backtracking. FAIL
12) ..... Goal B. No more rules to deduce it. Backtracking. FAIL
13) ..... Goal C. Find R9: [J] -> C. New goals:[J]
14) ..... Goal J. No rules to deduce it. Backtracking. FAIL
15) ..... Goal C. No more rules to deduce it. Backtracking. FAIL
16) .... Goal D. Find R6: [T] -> D. New goals:[T]
17) .... Goal T. Fact was given, DB:[T]
18) .... Goal D. New Fact added, DB:[T, D], Backtracking. OK
19) .... Goal C. Find R3: [B] -> C. New goals:[B]
20) ..... Goal B. Find R4: [A] -> B. New goals:[A]
21) ..... Goal A. Find R5: [D] -> A. New goals:[D]
22) ..... Goal D. Fact was deduced, DB:[T, D]
23) ..... Goal A. New Fact added, DB:[A, T, D], Backtracking. OK
24) ..... Goal B. New Fact added, DB:[A, B, T, D], Backtracking. OK
25) .... Goal C. New Fact added, DB:[A, B, C, T, D], Backtracking. OK
26) .. Goal Z. New Fact added, DB:[A, B, C, T, D, Z], Backtracking. OK

PART 3. Results
1) Z was deduced
2) Path: [R6, R5, R4, R3, R1]

Semantinis grafas



3.3.3 Devynios produkcijos C, D

Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//3 testas. Devynios produkcijos C, D.
//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

```
PART 1. Data
        1) Rules:
                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) Given facts:
T

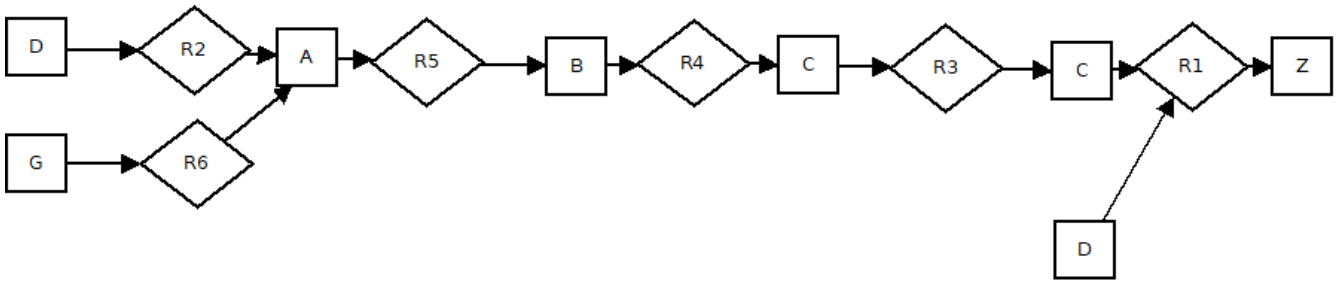
3) Goal:
Z

PART 2. Execution

1) .. Goal Z. Find R1: [C, D] -> Z. New goals:[C, D]
2) .... Goal C. Find R3: [B] -> C. New goals:[B]
3) ..... Goal B. Find R4: [A] -> B. New goals:[A]
4) ..... Goal A. Find R5: [D] -> A. New goals:[D]
5) ..... Goal D. Find R2: [C] -> D. New goals:[C]
6) ..... Goal C. Cycle detected, Backtracking. FAIL.
7) ..... Goal D. Find R6: [T] -> D. New goals:[T]
8) ..... Goal T. Fact was given, DB:[T]
9) ..... Goal D. New Fact added, DB:[T, D], Backtracking. OK
10) ..... Goal A. New Fact added, DB:[A, T, D], Backtracking. OK
11) ..... Goal B. New Fact added, DB:[A, B, T, D], Backtracking. OK
12) .... Goal C. New Fact added, DB:[A, B, C, T, D], Backtracking. OK
13) .. Goal D. Fact was deduced, DB:[A, B, C, T, D]
14) .. Goal Z. New Fact added, DB:[A, B, C, T, D, Z], Backtracking. OK

PART 3. Results
1) Z was deduced
2) Path: [R6, R5, R4, R3, R1]

Semantinis grafas



3.3.4 Penkios produkcijos ir praleistas potikslis

Įvestis

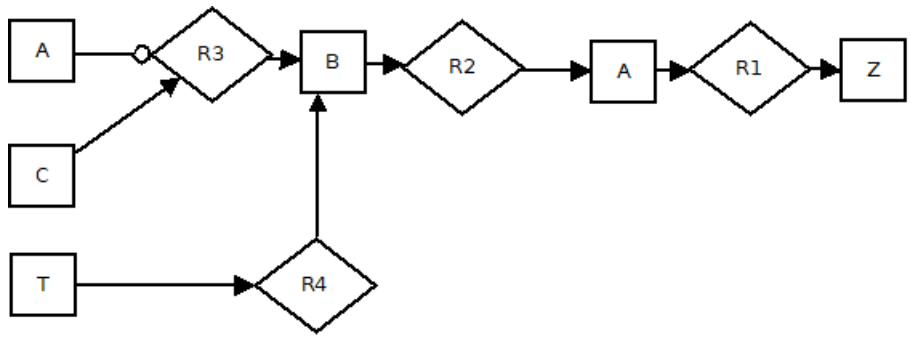
```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//4 testas. Penkios produkcijos ir praleistas potikslis.
//1) Taisyklės:
Z A   // R1: A -> Z
A B   // R2: B -> A
B A C // R3: A, C -> B
B T   // R4: T -> B
C T   // R5: T -> C

//2) Faktai:
T

//3) Tikslas:
Z
```

Išvestis

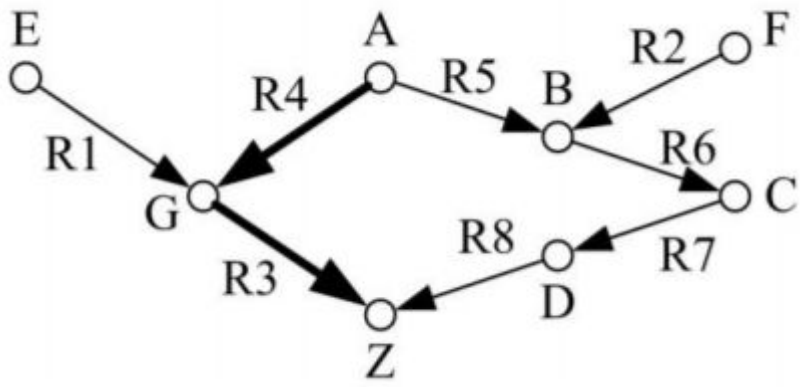
```
PART 1. Data
        1) Rules:
                R1: [A] -> Z
                R2: [B] -> A
                R3: [A, C] -> B
                R4: [T] -> B
                R5: [T] -> C

        2) Given facts:
                T

        3) Goal:
                Z

PART 2. Execution

   1) .. Goal Z. Find R1: [A] -> Z. New goals:[A]
   2) .... Goal A. Find R2: [B] -> A. New goals:[B]
   3) ..... Goal B. Find R3: [A, C] -> B. New goals:[A, C]
   4) ..... Goal A. Cycle detected, Backtracking. FAIL.
   5) ..... Goal B. Find R4: [T] -> B. New goals:[T]
   6) ..... Goal T. Fact was given, DB:[T]
   7) ..... Goal B. New Fact added, DB:[B, T], Backtracking. OK
   8) .... Goal A. New Fact added, DB:[A, B, T], Backtracking. OK
   9) .. Goal Z. New Fact added, DB:[A, B, T, Z], Backtracking. OK

PART 3. Results
```

1) Z was deduced
2) Path: [R4, R2, R1]

Semantinis grafas



3.3.5 Grafas su trumpu keliu



Įvestis

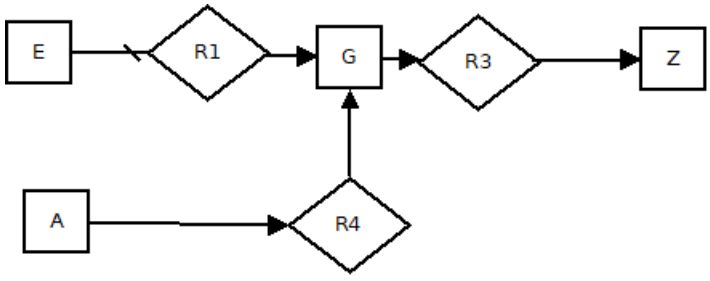
```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//5 testas. Grafas su trumpu keliu.
//1) Taisyklės:
G E // R1: E -> G
B F // R2: F -> B
Z G // R3: G -> Z
G A // R4: A -> G
B A // R5: A -> B
C B // R6: B -> C
D C // R7: C -> D
Z D // R8: D -> Z

//2) Faktai:
A
```

```
//3) Tikslas:
Z
```

Išvestis

```
PART 1. Data
        1) Rules:
                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) Given facts:
                A

        3) Goal:
                Z

PART 2. Execution

   1) .. Goal Z. Find R3: [G] -> Z. New goals:[G]
   2) .... Goal G. Find R1: [E] -> G. New goals:[E]
   3) ..... Goal E. No rules to deduce it. Backtracking. FAIL
   4) .... Goal G. Find R4: [A] -> G. New goals:[A]
   5) .... Goal A. Fact was given, DB:[A]
   6) .... Goal G. New Fact added, DB:[A, G], Backtracking. OK
   7) .. Goal Z. New Fact added, DB:[A, G, Z], Backtracking. OK

PART 3. Results
        1) Z was deduced
        2) Path: [R4, R3]
```

Semantinis grafas



3.3.6 Grafas su ilgu keliu

Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//6 testas. Grafas su ilgu keliu.
//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
```

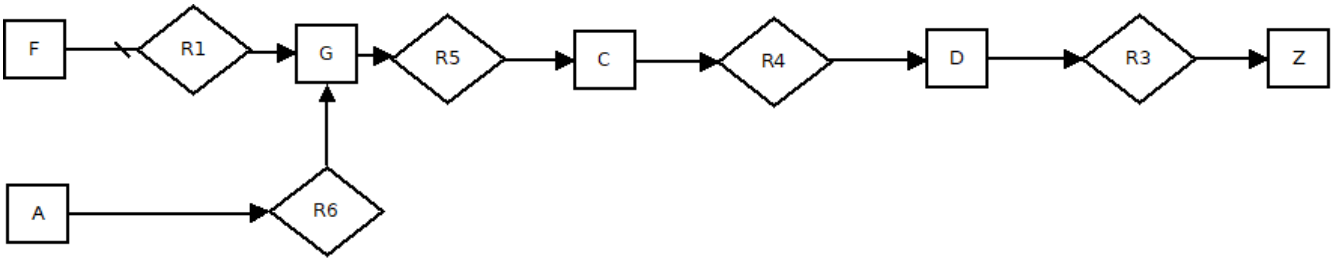
Išvestis

```
PART 1. Data
        1) Rules:
                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) Given facts:
                A

        3) Goal:
                Z

PART 2. Execution
```

1) .. Goal Z. Find R3: [D] -> Z. New goals:[D]
2) .... Goal D. Find R4: [C] -> D. New goals:[C]
3) ..... Goal C. Find R5: [B] -> C. New goals:[B]
4) ..... Goal B. Find R1: [F] -> B. New goals:[F]
5) ..... Goal F. No rules to deduce it. Backtracking. FAIL
6) ..... Goal B. Find R6: [A] -> B. New goals:[A]
7) ..... Goal A. Fact was given, DB:[A]
8) ..... Goal B. New Fact added, DB:[A, B], Backtracking. OK
9) ..... Goal C. New Fact added, DB:[A, B, C], Backtracking. OK
10) .... Goal D. New Fact added, DB:[A, B, C, D], Backtracking. OK
11) .. Goal Z. New Fact added, DB:[A, B, C, D, Z], Backtracking. OK

PART 3. Results
1) Z was deduced
2) Path: [R6, R5, R4, R3]

Semantinis grafas



3.3.7 *Trys alternatyvūs keliai*

Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//7 testas. Trys alternatyvūs keliai.
//1) Taisyklės:
Z A // R1: A -> Z
Z B // R2: B -> Z
Z C // R3: C -> Z

//2) Faktai:
C

//3) Tikslas:
Z
```

Išvestis

PART 1. Data

1) Rules:

R1: [A] -> Z
R2: [B] -> Z
R3: [C] -> Z

2) Given facts:

C

3) Goal:

Z

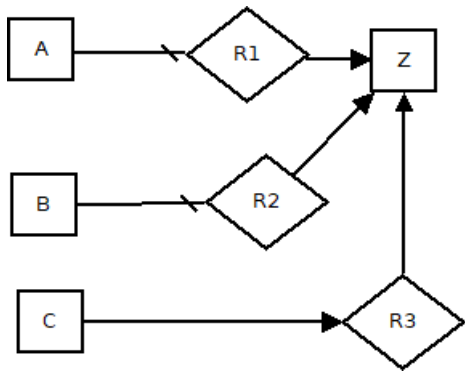
PART 2. Execution

1) .. Goal Z. Find R1: [A] -> Z. New goals:[A]
2) .... Goal A. No rules to deduce it. Backtracking. FAIL
3) .. Goal Z. Find R2: [B] -> Z. New goals:[B]
4) .... Goal B. No rules to deduce it. Backtracking. FAIL
5) .. Goal Z. Find R3: [C] -> Z. New goals:[C]
6) .. Goal C. Fact was given, DB:[C]
7) .. Goal Z. New Fact added, DB:[C, Z], Backtracking. OK

PART 3. Results

1) Z was deduced
2) Path: [R3]

Semantinis grafas



3.3.8 Trys alternatyvūs keliai ir nepasiekiamas tikslas

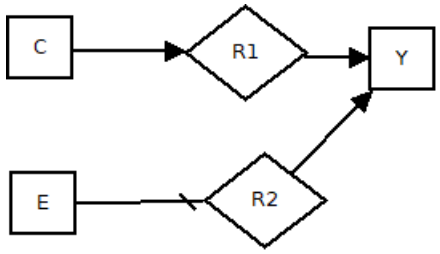
Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//8 testas. Trys alternatyvūs keliai ir nepasiekiamas tikslas.
//1) Taisyklės:
Z A D // R1: A, D -> Z
Z B D // R2: B, D -> Z
Z C D // R3: C, D -> Z
Y C E // R4: C, E -> Y

//2) Faktai:
C D

//3) Tikslas:
Y
```

Išvestis

```
PART 1. Data
        1) Rules:
                R1: [A, D] -> Z
                R2: [B, D] -> Z
                R3: [C, D] -> Z
                R4: [C, E] -> Y

        2) Given facts:
                C
                D

        3) Goal:
                Y

PART 2. Execution

   1) .. Goal Y. Find R4: [C, E] -> Y. New goals:[C, E]
   2) .. Goal C. Fact was given, DB:[C, D]
   3) .... Goal E. No rules to deduce it. Backtracking. FAIL
   4) .. Goal Y. No more rules to deduce it. Backtracking. FAIL

PART 3. Results
        1) Y was not deduced
```

Semantinis grafas



3.3.9 Tikslas tarp faktų

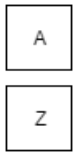
Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//9 testas. Tikslas tarp faktų
//1) Taisyklės:
Z A // R1: A -> Z

//2) Faktai:
A Z

//3) Tikslas:
Z
```

Išvestis

```
PART 1. Data
        1) Rules:
                R1: [A] -> Z

        2) Given facts:
                A
                Z

        3) Goal:
                Z

PART 2. Execution


PART 3. Results
        1) Z was given
```

Semantinis grafas



3.3.10 Negnevitsky pavyzdys (5 produkcijos, p.39)

Įvestis

```
//Studentas Laimonas Beniušis, informatikos: kompiuterių mokslo studijų programa, 4 kursas, 1 grupė
//10 testas. Negnevitsky pavyzdys (5 produkcijos, p.39)
//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
```

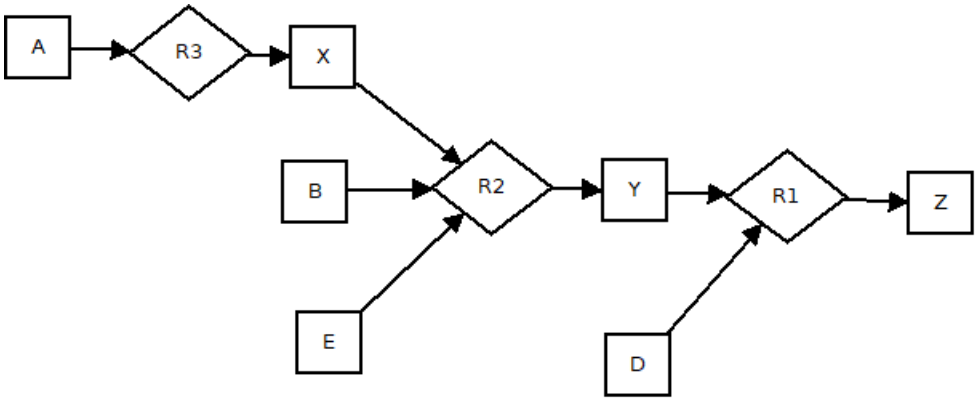
Išvestis

```
PART 1. Data
        1) Rules:
                R1: [Y, D] -> Z
                R2: [X, B, E] -> Y
                R3: [A] -> X
                R4: [C] -> L
                R5: [L, M] -> N

        2) Given facts:
                A
                B
                C
                D
                E

        3) Goal:
                Z

PART 2. Execution

   1) .. Goal Z. Find R1: [Y, D] -> Z. New goals:[Y, D]
   2) .... Goal Y. Find R2: [X, B, E] -> Y. New goals:[X, B, E]
   3) ..... Goal X. Find R3: [A] -> X. New goals:[A]
   4) ..... Goal A. Fact was given, DB:[A, B, C, D, E]
   5) ..... Goal X. New Fact added, DB:[A, B, C, D, E, X], Backtracking. OK
   6) .... Goal B. Fact was given, DB:[A, B, C, D, E]
```

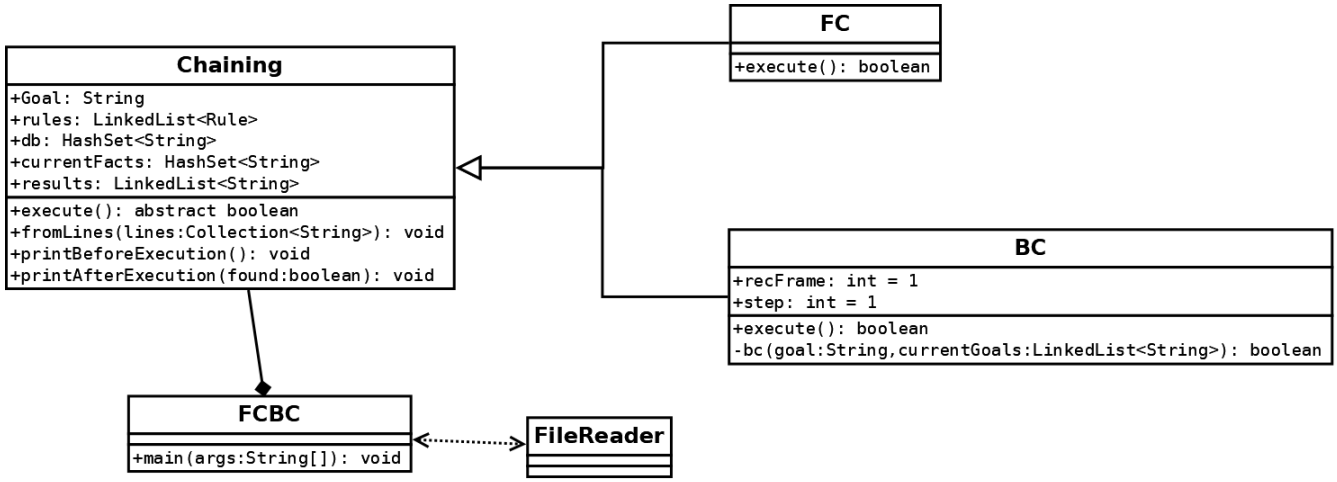
7) .... Goal E. Fact was given, DB:[A, B, C, D, E]
8) .... Goal Y. New Fact added, DB:[A, B, C, D, E, X, Y], Backtracking. OK
9) .. Goal D. Fact was given, DB:[A, B, C, D, E]
10) .. Goal Z. New Fact added, DB:[A, B, C, D, E, X, Y, Z], Backtracking. OK

PART 3. Results
1) Z was deduced
2) Path: [R3, R2, R1]

Semantinis grafas



4 UML klasių diagrama



5 Literatūros sąrašas

1. V. Čyras „Intelektualios sistemos“. http://www.mif.vu.lt/~cyras/AI/konspektas-intelektualios-sistemos.pdf

2. M. Negnevitsky „Artificial Intelligence. A Guide to Intelligent Systems“, Pearson Education Limited, Harlow, 2005.