VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Tiesioginis ir atbulinis išvedimas produkcijų sistemoje

Dalyko „Dirbtinis intelektas“ laboratorinio darbo, parašyto JAVA programavimo kalba, aprašymas

Darbą atliko 3 kurso 2 grupės studentas:
Egidijus Lukauskas

Vilnius – 2012
Versija: 1.0

Turinys

1. ĮVADAS

Šis darbas yra dalyko „Dirbtinis intelektas“ laboratorinio darbo savarankiška užduotis. Šiame darbe yra pateikiama programos, parašytos JAVA programavimo kalba ir simuliuojančios tiesioginio išvedimo metodo darbą, dokumentacija, darbo rezultatų pavyzdžiai ir aptarimas.

1.1. Produkcijų sistema

Produkcijų sistema yra vienas iš dirbtinio intelekto sistemų modelių. Ją sudaro trejetas:

1. globali duomenų bazė
2. produkcijų aibė
3. valdymo sistema

Globali duomenų bazė yra dirbtinio intelekto produkcijų sistemos naudojama duomenų struktūra. Ji turi būsenas, kurios yra keičiamos pritaikant produkcijas iš produkcijų aibės.

Produkcijų aibė yra sudaryta iš produkcijų $R_i : A \rightarrow B$, kur A yra aibė sąlygų, kurios turi būti patenkintos, kad galėtume pagal taisyklę R_i priimti rezultatus nurodytus aibėje B.

Valdymo sistema nurodo, kaip ir kurią produkciją iš produkcijų aibės parinkti taikymui. Valdymo sistema taip pat po kiekvienos produkcijos taikymo patikrina, ar globali duomenų bazės pasiekė galutinė būseną.

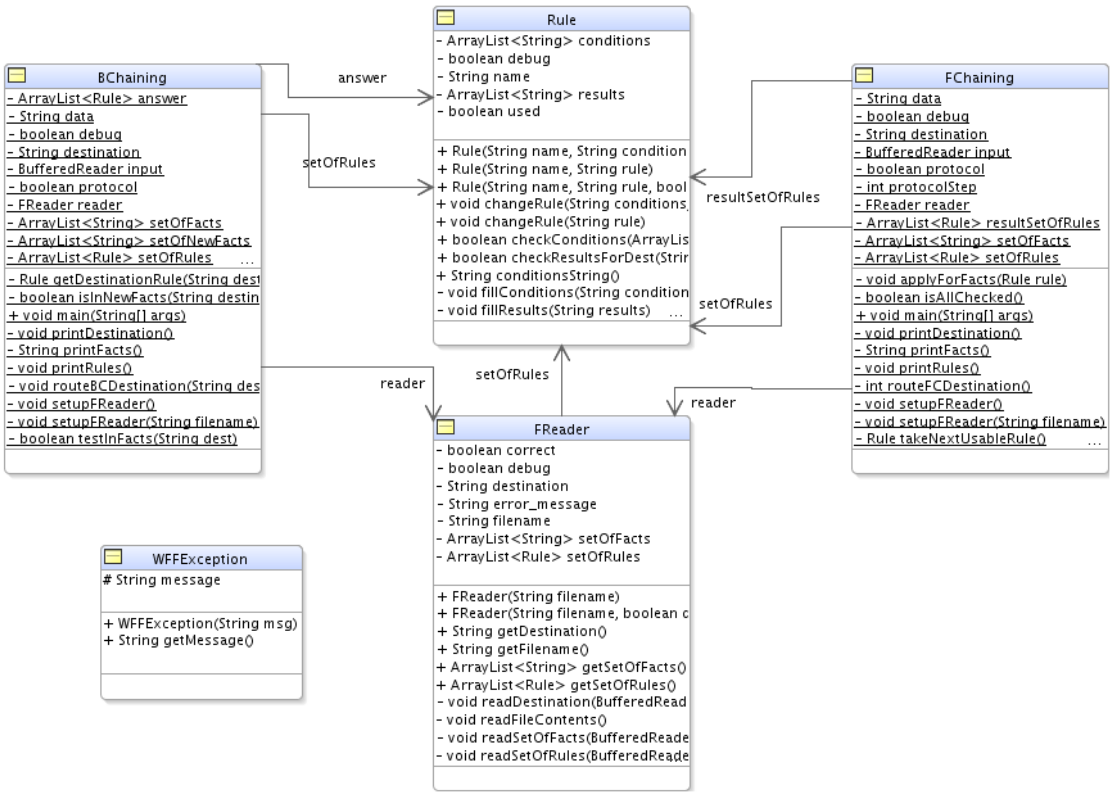
1.2. Tiesioginis išvedimas

Tiesioginis išvedimas (angliškai: forward chaining) yra metodas, skirtas gauti norimam tikslui, kai yra turimą produkcijų aibė ir pradinė globalios duomenų bazės būsena (pradiniai faktai). Tiesioginio išvedimo taikymo metu, produkcijų aibėje yra ieškoma tokia produkcija (taisyklė), kurią galima įvykdyti su tuo matu globalioje duomenų bazėje turimais faktais. Taisyklės pritaikymo metu gaunami nauji faktai ir jie įrašomi į globalią duomenų bazę pakeičiant jos būseną. Kiekviena produkcija gali būti taikoma tiek vieną kartą. Taip parinkinėjamos produkcijos tol, kol randamas tikslas (pasiekiama globalios duomenų bazės galutinė būsena) arba kol nebegalima pritaikyti nė vienos naujos tiaksyklės.

2. PROGRAMOS APRAŠAS

2.1. Programos struktūra

Beveik visas programos funkcionalumas realizuotas pagrindinėje klasėje $FChaining$, kuri naudoja pagalbinio paketo $utils$ klases $Rule$ ir $FReader$. Klasės tarpusavyje keičiasi klaidų pranešimu $WFFException$ (Wrong File Format Exception), kuris siunčiamas, kai nustatoma, jog duomenų failo struktūra yra netinkama.



2.1. pav.: Programos klasių diagrama (su paketų antraštėmis).

2.2. $Rule$ klasė

Klasė $Rule$ yra skirta atvaizduoti dirbtinio intelekto produkcijų sistemos produkcijos (taisyklės) objektą. Ji savyje saugo produkcijos vardą ir du sąrašus duomenų - sąlygų bei rezultatų. Tai yra pagalbinė klasė ir priklauso paketui $utils$.

Sąlygų sąrašas iš išorės nėra pasiekiamas, tačiau jį galima gauti kviečiant metodą $getConditions()$. Klasėje yra realizuotas metodas $checkConditions()$, kuris pagal pateiką globalios duomenų bazės faktų rinkinį nustato ar taisyklė yra taikytina, ar ne bei pateikią loginį rezultatą pagal tai.

Rezultatų sąrašas taip pat nėra tiesiogiai pasiekiamas iš išorės, tačiau jį galima gauti kviečiant metodą $getResults()$. Rezultatų aibė yra naudojama metode $isResutsInFacts()$. Šiam metodui reikia pateikti globalios duomenų bazės faktų rinkinį ir pagal jį yra patikrinama, ar produkcijos rezultas nėra dar įtrauktas į tą faktų rinkinį.

Taip pat klasėje yra saugoma būsena apie produkcijos panaudojimą, kurią galima pasiekti kviečiant metodą $isUsed()$.

2.2.1. Dalinis išeities tekstas

Daliniame išeities tekste pagalbinių metodų realizacijos nėra pateiktos.

```
/**
 * Rule object is designed to store the lists of conditions and results.
 * It represents the rule being used in production system for forward chaining.
 * @author Egidijus Lukauskas
 */
public class Rule {
  /*
   * Private items
   */
  // Rule data.
  private ArrayList<String> conditions = new ArrayList<String>();
  private ArrayList<String> results = new ArrayList<String>();
  // Default values.
  private boolean debug = false;
  private String name = "";
  private boolean used = false;

  /*
   * Constructors
   */
  public Rule(String name, String rule);
  public Rule(String name, String rule, boolean debug);
  public Rule(String name, String conditions, String results);

  /*
   * Public methods
   */

  /**
   * Iterates all the conditions of this rule and test if rule can be applied
   * by looking for these conditions in set of facts given.
   * @param setOfFacts  Set of facts to test if the conditions are met
   * @return  boolean if rule can be applied
```

```
34      */
35     public boolean checkConditions(ArrayList<String> setOfFacts) {
36       boolean ruleApplies = true;
37       boolean foundForCondition; int i;
38       for(String cond : this.conditions) {
39         // Take every condition form list and look for it in set of facts.
40         foundForCondition = false;
41         i = 0;
42         while(!foundForCondition && (setOfFacts.size() > i)) {
43           if(setOfFacts.get(i++).equals(cond)) {
44             // Condition is found in facts.
45             foundForCondition = true;
46           }
47         }
48         // If all conditions are found rule applies.
49         ruleApplies = ruleApplies && foundForCondition;
50       }
51       return ruleApplies;
52     }
53
54     /**
55       * Checks results for given destination.
56       * @param destination   String with destination
57       * @return boolean
58       */
59     public boolean checkResultsForDest(String destination) {
60       boolean ruleApplies = false;
61       for(String result : this.results) {
62         if(result.equals(destination)){
63           ruleApplies = true;
64         }
65       }
66       return ruleApplies;
67     }
68
69     /**
70      * Test if rule's results are already in set of facts given.
71      * @param setOfFacts  Set of facts to rest if results are in it
72      * @return  boolean if all the results are in set of facts
73      */
74     public boolean isResultsInFacts(ArrayList<String> setOfFacts) {
75       boolean isInFacts = true;
76       boolean foundForResult; int i;
```

```
77      // Take every result from list and look for it in set of facts.
78      for(String result : this.results) {
79        foundForResult = false;
80        i = 0;
81        while(!foundForResult && (setOfFacts.size() > i)) {
82          if(setOfFacts.get(i++).equals(result)) {
83            // Result is found in facts.
84            foundForResult = true;
85          }
86        }
87        // If all the results are found rule is in facts.
88        isInFacts = isInFacts && foundForResult;
89      }
90      return isInFacts;
91    }
92
93    /*
94     * Setters and getters.
95     */
96    public String getName();
97    public void setName(String name);
98    public boolean isUsed();
99    public void setUsed(boolean used);
100   public String toString();
101   public String resultsString();
102   public String conditionsString();
103   public ArrayList<String> getResults();
104   public ArrayList<String> getConditions();
105
106   /*
107    * Wrappers
108    */
109   private void parseRuleString(String rule);
110
111   /*
112    * Utility methods
113    */
114   private void fillConditions(String conditions);
115   private void fillResults(String results);
116
117   /*
118    * Extra functionality
119    */
```

```
120
121     /**
122      * Change current rule's data with other.
123      * @param rule  Rule string from which to read new data
124      */
125     public void changeRule(String rule) {
126       this.conditions = new ArrayList<String>();
127       this.results = new ArrayList<String>();
128       this.parseRuleString(rule);
129     }
130
131     /**
132      * Change current rule's data with other.
133      * @param conditions  String for conditions to be parsed
134      * @param results    String for results to be parsed
135      */
136     public void changeRule(String conditions, String results);
137 }
```

2.3. *FReader* klasė

Pagalbinė klasė, esanti pakete *utils*. Ši klasė paruošia pateiktą duomenų failą skaitymui, patikrina jo korektiškumą ir, jeigu failas korektiškas, inicializavimo metu nuskaito produkcijų sistemos duomenis. Esant nekorektiškam failui, kūrimo metu perduoda klaidos pranešimą $WFFException$ su klaidos žinute. Pagal nutylėjimą duomenų failu laikomas „produkcijos.txt“.

Failo skaitymas vykdomas vos pradėjus objekto kūrimą, todėl sukūrus $FReader$ objektą, jame jau yra surinkta produkcijų sistemos informacija: produkcijų ir pradinių globalios duomenų bazės faktų aibės bei galutinis tikslas. Ši infomracija yra saugoma ir nekeičiama iki kol nurodomas naujas failas arba objektas sunakinamas. Todėl duomenis galima pasiekti bet kuriuo metu po objekto sukūrimo. Visi šie duomenys iš išorės tiesiogiai nėra pasiekiami, tačiau jų reikšmes galima gauti kviečiant atitinkamai metodus $getSetOfRules()$, $getSetOfFacts()$ ir $getDestination()$.

2.3.1. Reikalavimai duomenų failui

1. Failas turi prasidėti autoriaus vardu ir pavarde.
2. Antra failo eilutė privalo būti: „1. Taisyklių aibė“.
3. Sekančiose eilutėse pateikiama produkcijų aibė (1 produkcija eilutėje). Produkcijose galima naudoti komentarus, juos pradedant „#“ simboliu.
4. Po produkcijų aibos paliekama tuščia eilutė.
5. Sekanti eilutė privalo būti: „2. Faktai“.
6. Kitoje eilutėje nurodoma faktų aibė. po jos paliekama tuščia eilutė.
7. Sekanti eilutė turi būti: „3. Tikslas“.

8. kitoje eilutėje nurodomas vienas simbolis - galutinis tikslas.

Duomenų failo pavyzdys:

```
Egidijus Lukauskas
1. Taisyklių aibė
LA                              # R1: A -> L
KL                              # R2: L -> K
AD                              # R3: D -> A
MD                              # R4: D -> M
ZFB                             # R5: F, B -> Z
FCD                             # R6: C, D -> F
DA                              # R7: A -> D

2. Faktai
ABC

3. Tikslas
Z

Failo pabaiga.
```

2.3.2. Dalinis išeities tekstas

```
/**
 * File reader object is designed to read from file the production system for
 * forward chaining and keep the lists of conditions, results and the
 * destination itself.
 * @author Egidijus Lukauskas
 */
public class FReader {

  // Default values.
  private String filename = "produkcijos.txt";
  private boolean correct = true;
  private String error_message = "";
  private boolean debug = false;

  // Production system's data read from file
  private ArrayList<Rule> setOfRules = new ArrayList<Rule>();
  private ArrayList<String> setOfFacts = new ArrayList<String>();
  private String destination;

  /*
   * Constructors
```

```
22      */
23     public FReader(String filename) throws WFFException;
24     public FReader(String filename, boolean debug) throws WFFException;
25
26     /*
27      * Private methods.
28      */
29     /**
30      * Prepares file to be read, test if author is correct and reads the data
31      * from file.
32      */
33     private void readFileContents() {
34       // Preparing file to read.
35       DataInputStream fileStream;
36           try {
37         fileStream = new DataInputStream(
38                     new FileInputStream(this.filename));
39       } catch (FileNotFoundException e) {
40         fileStream = null;
41         this.error_message = "Failas, pavadinimu „" +
42                     this.filename + "", nerastas.";
43         this.correct = false;
44       }
45           // File exists and is ready to be read. Starting reading.
46           if(fileStream != null) {
47             BufferedReader file = new BufferedReader(
48                         new InputStreamReader(fileStream));
49             try {
50               // Test if file's author is correct.
51               if((file.readLine()).trim().equals("Egidijus Lukauskas")) {
52                 // Read production system's data from file.
53                 this.readSetOfRules(file);
54                 this.readSetOfFacts(file);
55                 this.readDestination(file);
56               } else {
57                 // Incorrect file author.
58                 this.error_message = "Failas ne to autoriaus.";
59                 this.correct = false;
60               }
61             } catch(Exception e) {}
62           }
63     }
64
```

```
65    /**
66     * Tests if file structure for set of rules is correct. If so, reads the
67     * data from file.
68     * @param file      BufferedReader object where to read from
69     * @throws IOException  Input-Output Exception.
70     */
71    private void readSetOfRules(BufferedReader file) throws IOException {
72      int counter = 0; // Rule number counter.
73      String line;
74      String rule_expression;
75      // Test if structure is correct.
76      if((line = file.readLine()).trim().equals("1. Taisykliu aibe")) {
77        while (!(line = file.readLine()).trim().equals("")) {
78          // Remove unnecessary symbols from file.
79          rule_expression = line.replaceAll(" ", "").trim();
80          rule_expression = rule_expression.replaceAll("\t", "");
81          if(rule_expression.contains("#")) {
82            // Split before comments.
83            rule_expression = rule_expression.split("#")[0];
84          }
85          // Add newly read rule to set.
86          this.setOfRules.add(new Rule("R"+(++counter), rule_expression, this.debug));
87        }
88      } else {
89        // File structure incorrect.
90        this.error_message = "Taisyklių aibė nėra nurodyta taisyklingai.";
91          this.correct = false;
92      }
93    }
94
95    /**
96     * Tests if file structure for set of facts is correct. If so, reads the
97     * data from file.
98     * @param file      BufferedReader object where to read from
99     * @throws IOException  Input-Output Exception.
100    */
101   private void readSetOfFacts(BufferedReader file) throws IOException {
102     String line;
103     String facts_line;
104     // Test if structure is correct.
105     if((line = file.readLine()).trim().equals("2. Faktai")) {
106       while (!((line = file.readLine()).trim().equals(""))) {
107         // Remove unnecessary symbols from file.
```

```
108         facts_line = line.replaceAll(" ", "").trim();
109         facts_line = facts_line.replaceAll("\t", "").trim();
110         // Take every fact one by one.
111         for (int i=0;i<facts_line.length();i++) {
112           // Add the new fact to set of facts.
113           this.setOfFacts.add(facts_line.substring(i, i+1));
114           if(debug) {
115             System.out.println("Pridetas naujas faktas: "+
116                 facts_line.substring(i, i+1));
117           }
118         }
119       }
120     } else {
121       // File structure incorrect.
122       this.error_message = "Faktų aibė nėra nurodyta taisyklingai.";
123         this.correct = false;
124     }
125   }
126 
127   /**
128    * Tests if file structure for destination is correct. If so, reads the
129    * data from file.
130    * @param file      BufferedReader object where to read from
131    * @throws IOException  Input-Output Exception.
132    */
133   private void readDestination(BufferedReader file) throws IOException {
134     String line;
135     String dest;
136     // Test if structure is correct.
137     if((line = file.readLine()).trim().equals("3. Tikslas")) {
138       // Test if there is a destination.
139       if((line = file.readLine()).length() > 0) {
140         // Get the destination symbol.
141         dest = line.substring(0,1);
142         this.destination = dest;
143         if(debug) {
144           System.out.println("Nustatytas tikslas: "+this.destination);
145         }
146       } else {
147         // There is no destination. Wrong structure.
148         this.error_message = "Tikslas nėra nurodytas taisyklingai.";
149           this.correct = false;
150       }
```

```
151       } else {
152         // File structure incorrect
153         this.error_message = "Tikslas nėra nurodytas taisyklingai.";
154           this.correct = false;
155       }
156     }
157
158     /*
159      * Setters and getters.
160      */
161     public ArrayList<Rule> getSetOfRules();
162     public ArrayList<String> getSetOfFacts();
163     public String getDestination();
164     public String getFilename();
165     public void setFilename(String filename);
166 }
```

2.4. *FChaining* klasė

Pagrindinė klasė, kurios veiklos tikslas yra simuliuoti tiesioginio išvedimo dirbtinio intelekto produkcijų sistemoje metodo veikimą. Ji tam naudoja pakete *utils* esančias pagalbines klases *Rule* ir *FReader*. Ši klasė yra pilnai statinė bei skirta tik algoritmo realizacijai bei pagalbinių klasių procesų valdymui.

Pagrindinė funkcija yra $routeFCDestination()$, kuri ir valdo pagrindinį tiesionio išvedimo algoritmo įgyvendinimą. Ji tiesiogiai įgyvendina 2, 5 ir 6 pseudokodo žinksnius. Siekiant supaprastinti funkcijos skaitomumą bei laikantis tinkamo kodavimo stiliaus principų, dalis algoritmo funkcionalumo buvo iškelta į keturias pagalbines funkcijas: $testForDestination()$, $isAllChecked()$, $takeNextUsableRule()$, $applyForFacts()$.

Funkcija $testForDestination()$ tikrina ar tarp globalios duomenų bazės faktų dar nėra galutinio tikslo. Jeigu rezultatas yra „-1“, tai tiklas yra nurodytas tarp faktų.

Funkcija $isAllChecked$ tikrina ar jau visos produkcijos buvo pritaikytos. Ji naudojama tam, kad žinotumėme, jog nėra kitų taikytinų taisyklių ir algoritmo darbas baigtas.

Funkcija $takeNextUsableRule()$ iš produkcijų aibės parenka kitą taisyklę, kurią būtų galima pritaikyti ir grąžina *Rule* objektą. Jeigu nebėra nei vienos taikytinos taisyklės, grąžinamas *null*. Šioje funkcijoje yra realizuoti 3 ir 4 pseudokodo žingsniai.

Funkcija $applyForFacts()$ paima taikomos taisyklės rezultatų aibė ir ją prijungia prie globalios duomenų bazės turimų faktų aibės.

2.4.1. Dalinis išeities tekstas

```
1 /**
2  * FChaining is class, which simulates reading production system for forward
```

```
3   * chaining and it's algorithm itself.
4   * @author Egidijus Lukauskas
5   */
6  public class FChaining {
7
8    private static FReader reader;
9    private static BufferedReader input =
10         new BufferedReader(new InputStreamReader(System.in));
11   private static String data = "";
12
13   // Printing purpose variables.
14   private static boolean debug = false;
15   private static boolean protocol = true;
16   private static int protocolStep = 1;
17
18   // Program's input-output data.
19   private static ArrayList<Rule> setOfRules = new ArrayList<Rule>();
20   private static ArrayList<String> setOfFacts = new ArrayList<String>();
21   private static String destination;
22   private static ArrayList<Rule> resultSetOfRules = new ArrayList<Rule>();
23
24   /**
25    * Main simulation function to control all the functional steps.
26    * @param args  command line arguments
27    */
28   public static void main(String[] args) {
29     System.out.println("*** Egidijaus Lukausko FChaining programa"+
30                        " pradeda darbą.");
31     try {
32       // Initializing reader object.
33       if(args.length == 0) {
34         setupFReader();
35       } else {
36         setupFReader(args[0]);
37       }
38     } catch (WFFException wffe) {
39       // Wrong File Format Exception occurred.
40       reader = null;
41       System.out.println(wffe.getMessage());
42     }
43
44     if(reader != null) {
45       if(debug)
```

```
46            System.out.println("*** Failo skaitymas baigtas.\n");
47
48          // Get data from recently read file to simulator.
49          setOfRules = reader.getSetOfRules();
50          setOfFacts = reader.getSetOfFacts();
51          destination = reader.getDestination();
52
53          // Print out production system.
54          System.out.println("Produkcijos:\n");
55          printRules();
56          System.out.println("\nDuomenys:");
57          printFacts();
58          System.out.println("\nTikslas:");
59          printDestination();
60          if(protocol)
61            System.out.println("\n[Programos darbo eiga]");
62
63          // Call solver function to get the result status.
64          int solution = routeFCDestination();
65
66          if(protocol){
67            if(solution == -1) {
68              // Destination already in facts.
69              System.out.println((protocolStep++)+
70                  ". Galutinis tikslas jau pateiktas tarp faktų.");
71              System.out.println(
72                  "Uždavinio sprendimas: {}.");
73            } else if((solution == -2) || (solution == 0)) {
74              // Destination not reachable.
75              System.out.println((protocolStep++)+
76                  ". Galutinio tikslo pasiekti neįmanoma.");
77              System.out.println(
78                  "Uždavinio sprendimas duotoje produkcijų sistemoje neegzistuoja.");
79            } else if(solution == 1) {
80              // Destination reached.
81              System.out.println((protocolStep++)+
82                  ". Tikslas su duota produkciju sistema pasiektas!");
83              System.out.print("Uždavinio sprendimas: {");
84              String delimiter = "";
85              // Print set of results.
86              for(Rule result : resultSetOfRules) {
87                System.out.print(delimiter+result.getName());
88                delimiter = "; ";
```

```
89            }
90            System.out.print("}.\n");
91          }
92          System.out.println("[Programos darbo eigos pabaiga]");
93        }
94      };
95
96      // System has finished it's calculations.
97      System.out.println("\n*** Programa baigia darba.");
98    }
99
100   /**
101    * Gets a rule to apply, applies it for facts and result array.
102    * Depending on results decides what is the result status and returns it.
103    *
104    * @return restinationReached  number for destination status: 1 - reached,
105    *                  -2 & 0 - not reachable, -1 - already in facts.
106    */
107   private static int routeFCDestination() {
108     Rule rule;
109     // Test if destination already in facts.
110     int destinationReached = testForDestination();
111     while((destinationReached == 0) && !isAllChecked()) { // #2
112       // Get next rule to apply.
113       rule = takeNextUsableRule();                        // #3, #4
114       if(rule != null) {
115         applyForFacts(rule);    // Add new applied facts. #5
116         resultSetOfRules.add(rule);  // Add new Rule to the set of results. #6
117         if(testForDestination() == -1)
118           destinationReached = 1; // Destination Reached.
119       } else {
120         // Destination not reachable.
121         destinationReached = -2;
122       }
123     }
124     return destinationReached;
125   }
126
127   /**
128    * Add rules results as new facts and sets the rule as already used.
129    * If protocol enabled prints the stack trace.
130    *
131    * @param rule  Rule object which should be applied for facts.
```

```
132     */
133    private static void applyForFacts(Rule rule);
134    private static Rule takeNextUsableRule();
135
136    /**
137     * Looks in set of facts to see if there is a destination in it already.
138     *
139     * @return Number with destination being if facts status: -1 - in facts,
140     *                                  0 - not in facts.
141     */
142    private static int testForDestination();
143    private static boolean isAllChecked();
144
145    private static void printRules();
146    private static void printFacts();
147    private static void printDestination();
148    private static void setupFReader(String filename) throws WFFException;
149    private static void setupFReader() throws WFFException;
150 }
```

2.5. *BChaining* klasė

Pagrindinė klasė, kurios veiklos tikslas yra simuliuoti atbulinio išvedimo dirbtinio intelekto produkcijų sistemoje metodo veikimą. Ji tam naudoja pakete *utils* esančias pagalbines klases *Rule* ir *FReader*. Ši klasė yra pilnai statinė bei skirta tik algoritmo realizacijai bei pagalbinių klasių procesų valdymui.

Pagrindinė funkcija yra *routeBCDestination*(), kuri ir valdo pagrindinį atbulinio išvedimo algoritmo įgyvendinimą. Ji tiesiogiai įgyvendina 2, 5 ir 6 pseudokodo žinksnius.Siekiant supaprastinti funkcijos skaitomumą bei laikantis tinkamo kodavimo stiliaus principų, dalis algoritmo funkcionalumo buvo iškelta į tris pagalbines funkcijas: *getDestinationRule*(), *isInNewFacts*(), *testInFacts*().

Funkcija *isInNewFacts*() tikrina ar nurodytas tikslas yra tarp naujai išvestų faktų. Jeigu taip, tada grąžinama reikšmė *true*.

Funkcija *testInFacts* tikrina ar nurodytas tikslas yra tarp pradinių faktų. Jeigu taip, tada grąžinama reikšmė *true*.

Funkcija *getDestinationRule*() iš produkcijų aibės parenka kitą taisyklę, kurią būtų galima pritaikyti ir turi nurodytą rezultatą, bei grąžina *Rule* objektą. Jeigu nebėra nei vienos taikytinos taisyklės, grąžinamas *null*.

2.5.1. Dalinis išeities tekstas

```
1  /**
2   * BChaining is class, which simulates reading production system for backward
```

```
3   * chaining and it's algorithm itself.
4   * @author Egidijus Lukauskas
5   */
6  public class BChaining {
7
8    private static FReader reader;
9    private static BufferedReader input =
10         new BufferedReader(new InputStreamReader(System.in));
11   private static String data = "";
12
13   // Printing purpose variables.
14   private static boolean debug = false;
15   private static boolean protocol = true;
16
17   // Program's input-output data.
18   private static ArrayList<Rule> setOfRules = new ArrayList<Rule>();
19   private static ArrayList<String> setOfFacts = new ArrayList<String>();
20   /* #2 */
21   private static ArrayList<String> setOfNewFacts = new ArrayList<String>();
22   private static String destination;
23   private static ArrayList<Rule> answer = new ArrayList<Rule>(); /* #1 */
24
25   /**
26    * Main simulation function to control all the functional steps.
27    * @param args  command line arguments
28    */
29   public static void main(String[] args) {
30     System.out.println("*** Egidijaus Lukausko BChaining"
31                 + " programa pradeda darbą.");
32     try {
33       // Initializing reader object.
34       if(args.length == 0) {
35         setupFReader();
36       } else {
37         setupFReader(args[0]);
38       }
39     } catch (WFFException wffe) {
40       // Wrong File Format Exception occurred.
41       reader = null;
42       System.out.println(wffe.getMessage());
43     }
44
45     if(reader != null) {
```

```
46          if(debug)
47            System.out.println("*** Failo skaitymas baigtas.\n");
48
49          // Get data from recently read file to simulator.
50          setOfRules = reader.getSetOfRules();
51          setOfFacts = reader.getSetOfFacts();
52          destination = reader.getDestination();
53
54          // Print out production system.
55          System.out.println("Produkcijos:\n");
56          printRules();
57          System.out.println("\nDuomenys:");
58          System.out.println(printFacts());
59          System.out.println("\nTikslas:");
60          printDestination();
61          if(protocol)
62            System.out.println("\n[Programos darbo eiga]");
63
64          routeBCDestination(destination,"",1);
65                System.out.println("[Programos darbo eigos pabaiga]\n");
66
67          if(protocol){
68            if(answer.isEmpty() && setOfNewFacts.isEmpty()) {
69              // Destination already in facts.
70              System.out.println("Galutinis tikslas jau pateiktas"
71                                 + " tarp faktų.");
72              System.out.println("Uždavinio sprendimas: {}.");
73            } else if(answer.isEmpty() && !setOfNewFacts.isEmpty()) {
74              // Destination not reachable.
75              System.out.println("Galutinio tikslo pasiekti"
76                                 + " neįmanoma.");
77              System.out.println("Uždavinio sprendimas duotoje produkcijų"
78                                 + " sistemoje neegzistuoja.");
79            } else if(!answer.isEmpty() && !setOfNewFacts.isEmpty()) {
80              // Destination reached.
81              System.out.println("Tikslas su duota produkcijų"
82                                 + " sistema pasiektas!");
83
84                          System.out.print("Uždavinio sprendimas: {");
85                          String delimiter = "";
86                          // Print set of results.
87                          for(Rule result : answer) {
88                              System.out.print(delimiter+result.getName());
```

```
89                          delimiter = "; ";
90                      }
91                      System.out.print("}.\n");
92                  }
93          }
94      }
95
96      // System has finished it's calculations.
97      System.out.println("\n*** Programa baigia darbą.");
98  }
99
100 /* #3 */
101 private static ArrayList<String> testedDestinations = new ArrayList<String>();
102
103   /*
104    * Backward Chaining algorythm function. Does not return anything, but
105    * changes the elements of static global variable ArrayList<Rule> answer.
106    */
107   private static void routeBCDestination(String destination, String spaces,
108       int level) {                                                    /* #4 */
109       ArrayList<Rule> rulesToApply = new ArrayList<Rule>();           /* #5 */
110       int sublevel = 0;
111       /* Test if the given destination is not in original facts */
112       if(!testInFacts(destination)){                                  /* #6 */
113
114           /* Get all the rules with given destination */
115           Rule ruleWithDestination;                                   /* #7 */
116           while((ruleWithDestination = getDestinationRule(destination)) != null)
117               rulesToApply.add(ruleWithDestination);                  /* #8-9 */
118
119           /*  Test for DeadEnd */
120           if(rulesToApply.isEmpty())
121               System.out.println(spaces+level+
122                       ". Aklavietė. Nėra taisyklių šio fakto išvedimui.");
123
124           /* Test for loop */
125           if(testedDestinations.contains(destination)) {              /* #11 */
126               System.out.println(spaces+"   Ciklas su tikslu: "+
127                       destination+" ------------------------");      /* #12 */
128               rulesToApply.clear();                                   /* #13 */
129           }
130
131           /* Try to apply every rule with given destination */
```

```
132             for(Rule rule : rulesToApply) {                          /* #15 */
133                 if(!isInNewFacts(destination)){                      /* #16 */
134                     sublevel++;
135                     System.out.println(spaces+level+". Einamas tikslas: "+
136                             destination+". Rasta taisyklė "+rule+
137                             ". Nauji tikslai: "+rule.conditionsString());
138 
139                     /*
140                       * Save destinations we already checked to
141                       * beware of loops
142                       */
143                     testedDestinations.add(rule.resultsString());  /* #17 */
144 
145                     /* Test every condition for the rule */
146                     for(String condition : rule.getConditions()) { /* #18 */
147                         /* If not have this as a fact already */
148                         if(!isInNewFacts(condition)) {             /* #19 */
149                             /* Recoursive call with new destination */
150                             routeBCDestination(condition,spaces+"    ",
151                                                level+1);          /* #20 */
152                         } else {
153                             /* Already in facts */
154                             System.out.println(spaces+"    "+(level+1)+
155                                     ". Tikslas: "+condition+
156                                     " jau išvestas anksčiau.");
157                         }
158                     }
159 
160                     /*
161                       * If all conditions are applied add new rule to the
162                       * results array
163                       */
164                     if(rule.checkConditions(setOfNewFacts)) {      /* #23 */
165                         setOfNewFacts.add(rule.resultsString());   /* #24 */
166                         answer.add(rule);                          /* #25 */
167                     }
168 
169                     /* Restore the rule we just checked */
170                     rule.setUsed(false);                            /* #27 */
171                     testedDestinations.remove(rule.resultsString());/* #28 */
172                 }
173             }
174         } else {
```

```
175                 /* Already in original facts */
176                 System.out.println(spaces+level+". Tikslas "+destination+
177                         " yra tarp faktų.");
178                 setOfNewFacts.add(destination);                         /* #32 */
179             }
180     }
181
182       /**
183        * Checks if given destination is in new facts list.
184        * @param destination   String with destination to check
185        * @return boolean
186        */
187     private static boolean isInNewFacts(String destination);
188
189       /**
190        * Checks if given destination is in original facts list.
191        * @param dest  String with destination to check
192        * @return boolean
193        */
194     private static boolean testInFacts(String dest);
195
196       /**
197        * Finds 1st unused rule with given destination.
198        * @param destination   String with rule restination we need.
199        * @return Rule object with given destination and isUsed() == false.
200        */
201     private static Rule getDestinationRule(String destination);
202     private static void printRules();
203     private static String printFacts();
204     private static void printDestination();
205     private static void setupFReader(String filename) throws WFFException;
206     private static void setupFReader() throws WFFException;
207 }
```

3. TIESIOGINIS IŠVEDIMAS

3.1. Pseudokodas

Rezultatas yra taisyklių, kurios buvo pritaikytos iki pasiekiant galutinę būseną, aibė. R žymi kintamąjį, nurodantį tuo metu nagrinėjamą produkciją. Išvados - produkcijos rezultatas. Faktais trumpiau vadiname einamojoje būsenoje esančios globalios duomenų bazės faktus.

1: $Rezultatas := ()$
2: **while** yra nepanaudotų taisyklių $\wedge$ tikslas nėra tarp faktų **do**
3: R :=nepanaudota taisyklė;
4: **if** R sąlygos yra tarp faktų $\wedge$ bent vienos R išvados nėra tarp faktų **then**
5: R išvadas pridedame prie faktų;
6: R pridedame į $Rezultatas$ galą;
7: **end if**
8: **end while**

3.2. Pavyzdžiai

3.2.1. Pirmas pavyzdys - 7 produkcijų pasiekiamas tikslas

Duomenų failas

```
1  Egidijus Lukauskas
2  1. Taisyklių aibė
3  LA                              # R1: A -> L
4  KL                              # R2: L -> K
5  AD                              # R3: D -> A
6  MD                              # R4: D -> M
7  ZFB                             # R5: F, B -> Z
8  FCD                             # R6: C, D -> F
9  DA                              # R7: A -> D
10
11 2. Faktai
12 ABC
13
14 3. Tikslas
15 Z
16
17 Failo pabaiga.
```

Duomenys

```
Produkcijos:

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

Duomenys:
A, B, C

Tikslas:
Z
```

Programos žingsniai

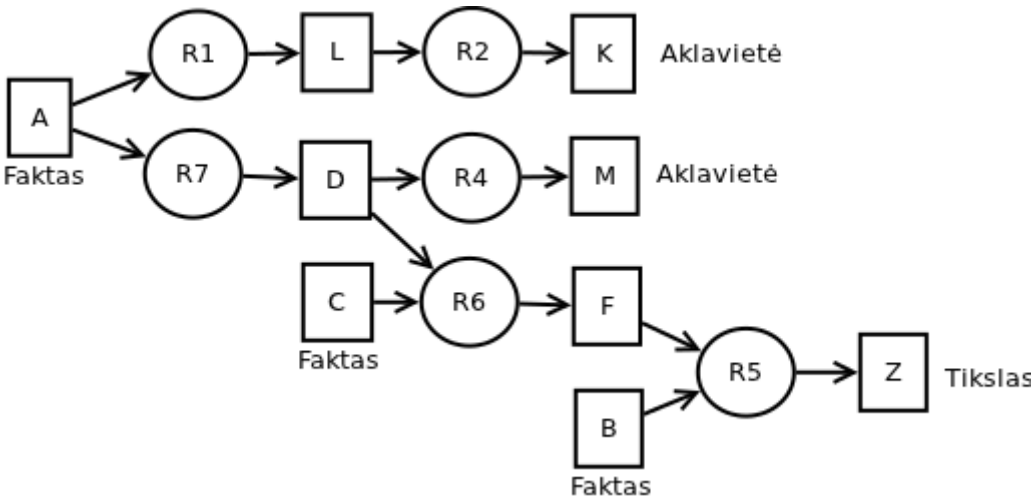
```
1. Taikoma taisyklė R1: A -> L. Prie faktų pridedamas naujas faktas: L.
Faktų aibė: {A, B, C, L}.
2. Taikoma taisyklė R2: L -> K. Prie faktų pridedamas naujas faktas: K.
Faktų aibė: {A, B, C, L, K}.
3. Taikoma taisyklė R7: A -> D. Prie faktų pridedamas naujas faktas: D.
Faktų aibė: {A, B, C, L, K, D}.
4. Taikoma taisyklė R4: D -> M. Prie faktų pridedamas naujas faktas: M.
Faktų aibė: {A, B, C, L, K, D, M}.
5. Taikoma taisyklė R6: C, D -> F. Prie faktų pridedamas naujas faktas: F.
Faktų aibė: {A, B, C, L, K, D, M, F}.
6. Taikoma taisyklė R5: F, B -> Z. Prie faktų pridedamas naujas faktas: Z.
Faktų aibė: {A, B, C, L, K, D, M, F, Z}.
7. Tikslas su duota produkcijų sistema pasiektas!
```

Rezultatas

```
Uždavinio sprendimas: {R1; R2; R7; R4; R6; R5}.
```

Semantinis gafas

Semantinis grafas pateiktas 3.1. paveiklėlyje.



3.1. pav.: 7 produkcijų su pasiekiamu tikslu sekmantinis grafas.

3.2.2. Antras pavyzdys - tikslas nepasiekiamas

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
EB                              # R1: B -> E
FB                              # R2: B -> F
BA                              # R3: A -> B
CB                              # R4: B -> C
DC                              # R5: C -> D

2. Faktai
A

3. Tikslas
Z

Failo pabaiga.
```

Įvestis

```
Produkcijos:

R1: B -> E
R2: B -> F
R3: A -> B
R4: B -> C
R5: C -> D

Duomenys:
```

```
A

Tikslas:
Z
```

Programos žingsniai

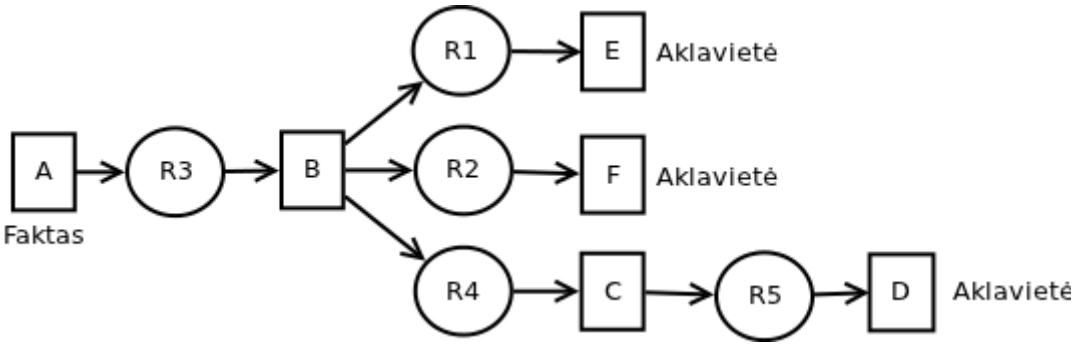
```
1. Taikoma taisyklė R3: A -> B. Prie faktų pridedamas naujas faktas: B.
Faktų aibė: {A, B}.
2. Taikoma taisyklė R1: B -> E. Prie faktų pridedamas naujas faktas: E.
Faktų aibė: {A, B, E}.
3. Taikoma taisyklė R2: B -> F. Prie faktų pridedamas naujas faktas: F.
Faktų aibė: {A, B, E, F}.
4. Taikoma taisyklė R4: B -> C. Prie faktų pridedamas naujas faktas: C.
Faktų aibė: {A, B, E, F, C}.
5. Taikoma taisyklė R5: C -> D. Prie faktų pridedamas naujas faktas: D.
Faktų aibė: {A, B, E, F, C, D}.
6. Galutinio tikslo pasiekti neįmanoma.
```

Rezultatas

```
Uždavinio sprendimas duotoje produkcijų sistemoje neegzistuoja.
```

Semantinis gafas

Semantinis grafas pateiktas 3.2. paveiklėlyje.



3.2. pav.: 5 produkcijų su nepasiekiamu tikslu semantinis grafas.

3.2.3. Trečias pavyzdys - tikslas tarp faktų

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
ZDC                             # R1: D, C -> Z
DC                              # R2: C -> D
CB                              # R3: B -> C
```

```
BA                              # R4: A -> B
AD                              # R5: D -> A
DT                              # R6: T -> D
AG                              # R7: G -> A
BH                              # R8: H -> B
CJ                              # R9: J -> C

2. Faktai
TZ

3. Tikslas
Z

Failo pabaiga.
```

Įvestis

```
Produkcijos:

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

Duomenys:
T, Z

Tikslas:
Z
```

Programos žingsniai

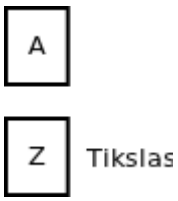
```
1. Galutinis tikslas jau pateiktas tarp faktų.
```

Rezultatas

```
Uždavinio sprendimas: {}.
```

Semantinis gafas

Semantinis grafas pateiktas 3.3. paveiklėlyje.



3.3. pav.: 9 produkcijų su tikslu tarp faktų semantinis grafas.

3.2.4. Ketvirtas pavyzdys - du išvedimo keliai

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
BA                              # R1: A -> B
ZG                              # R2: G -> Z
GA                              # R3: A -> G
ZD                              # R4: D -> Z
DC                              # R5: C -> D
CB                              # R6: B -> C

2. Faktai
A

3. Tikslas
Z

Failo pabaiga.
```

Įvestis

```
Produkcijos:

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

Duomenys:
A

Tikslas:
Z
```

Programos žingsniai

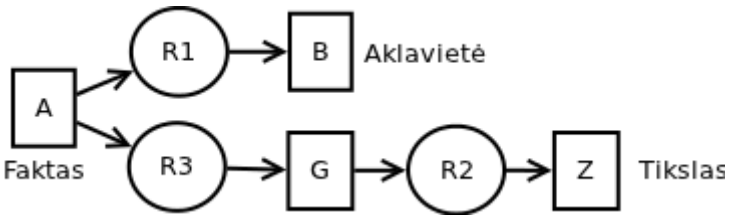
```
1. Taikoma taisyklė R1: A -> B. Prie faktų pridedamas naujas faktas: B.
Faktų aibė: {A, B}.
2. Taikoma taisyklė R3: A -> G. Prie faktų pridedamas naujas faktas: G.
Faktų aibė: {A, B, G}.
3. Taikoma taisyklė R2: G -> Z. Prie faktų pridedamas naujas faktas: Z.
Faktų aibė: {A, B, G, Z}.
4. Tikslas su duota produkcijų sistema pasiektas!
```

Rezultatas

```
Uždavinio sprendimas: {R1; R3; R2}.
```

Semantinis gafas

Semantinis grafas pateiktas 3.4. paveiklėlyje.



3.4. pav.: Dviejų išvedimo kelių semantinis grafas

3.2.5. Penktas pavyzdys - du išvedimo keliai, kita taisyklių tvarka

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
ZG                              # R1: G -> Z
BA                              # R2: A -> B
ZD                              # R3: D -> Z
DC                              # R4: C -> D
CB                              # R5: B -> C
GA                              # R6: A -> G

2. Faktai
A

3. Tikslas
Z

Failo pabaiga.
```

Įvestis

```
Produkcijos:

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

Duomenys:
A

Tikslas:
Z
```

Programos žingsniai

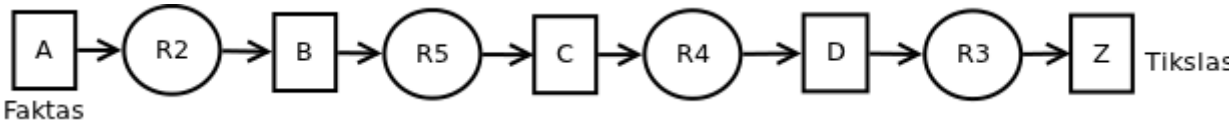
```
1. Taikoma taisyklė R2: A -> B. Prie faktų pridedamas naujas faktas: B.
Faktų aibė: {A, B}.
2. Taikoma taisyklė R5: B -> C. Prie faktų pridedamas naujas faktas: C.
Faktų aibė: {A, B, C}.
3. Taikoma taisyklė R4: C -> D. Prie faktų pridedamas naujas faktas: D.
Faktų aibė: {A, B, C, D}.
4. Taikoma taisyklė R3: D -> Z. Prie faktų pridedamas naujas faktas: Z.
Faktų aibė: {A, B, C, D, Z}.
5. Tikslas su duota produkcijų sistema pasiektas!
```

Rezultatas

```
Uždavinio sprendimas: {R2; R5; R4; R3}.
```

Semantinis gafas

Semantinis grafas pateiktas **??** paveiklėlyje.

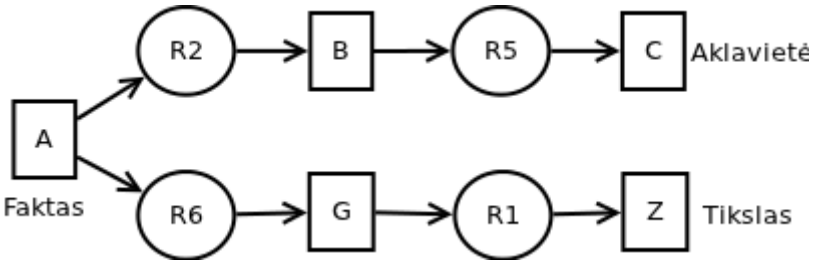


3.5. pav.: 6 produkcijų su dviem išvedimo keliais semantinis grafas.

Palyginimas su M. Negnevitsky realizacijos variantu

Pagal M. Negnevitsky sprendimo modelį, pirmiausia tiakomos $R2$ ir $R6$, nes jos yra konfliktuojančios tarpusavyje (angl. confict set). Toliau ieškoma taikomos taisyklės pradedant ne nuo

pradžių, o nuo $R3$. Sekanti taikoma randama $R5$, nes pritaikius $R2$, buvo gautas faktas B. Pritaikius $R5$, gaunamas naujas faktas C. Kadangi $R6$ jau buvo pritaikyta pirmame žingsnyje, tai taisyklių vėl ieškoma nuo pradžių. Taip randama $R1$, kurios rezultatas yra Z, t.y. galutinis tikslas. Algoritmas baigia darbą pateikdamas tokį rezultatą: $\{R2; R6; R5; R1\}$.



3.6. pav.: Dviejų išvedimo kelių semantinis grafas pagal M. Negnevitsky.

3.2.6. Šeštas pavyzdys - labirinto kelias nagrinėtas paskaitoje

Duomenų failas

```
1  Egidijus Lukauskas
2  1. Taisyklių aibė
3  @y
4  @t
5  as
6  sa
7  bs
8  sb
9  cs
10 sc
11 ds
12 sd
13 ea
14 ae
15 fe
16 ef
17 gf
18 fg
19 hg
20 gh
21 ih
22 hi
jh
hj
ki
ik
lk
kl
ml
lm
nm
mn
om
mo
pn
np
qo
oq
jq
qj
rq
qr
br
rb
uc
cu
vd
dv
wv
vw
xv
vx
zw
wz
tz
zt
yv
vy

2. Faktai
s

3. Tikslas
@

Failo pabaiga.
```

Įvestis

```
Produkcijos:

R1: t -> @
R2: y -> @
R3: s -> a
R4: a -> s
R5: s -> b
R6: b -> s
R7: s -> c
R8: c -> s
R9: s -> d
R10: d -> s
R11: a -> e
R12: e -> a
R13: e -> f
R14: f -> e
R15: f -> g
R16: g -> f
R17: g -> h
R18: h -> g
R19: h -> i
R20: i -> h
R21: h -> j
R22: j -> h
R23: i -> k
R24: k -> i
R25: k -> l
R26: l -> k
R27: l -> m
R28: m -> l
R29: m -> n
R30: n -> m
R31: m -> o
R32: o -> m
R33: n -> p
R34: p -> n
R35: o -> q
R36: q -> o
R37: q -> j
R38: j -> q
R39: q -> r
R40: r -> q
R41: r -> b
R42: b -> r
R43: c -> u
R44: u -> c
R45: d -> v
R46: v -> d
R47: v -> w
R48: w -> v
R49: v -> x
R50: x -> v
R51: w -> z
R52: z -> w
R53: z -> t
R54: t -> z
R55: v -> y
R56: y -> v

Duomenys:
s

Tikslas:
@
```

Programos žingsniai

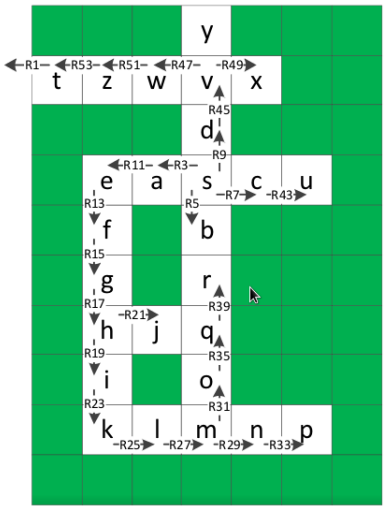
1. Taikoma taisyklė R3: s -> a. Prie faktų pridedamas naujas faktas: a.
2. Taikoma taisyklė R5: s -> b. Prie faktų pridedamas naujas faktas: b.
3. Taikoma taisyklė R7: s -> c. Prie faktų pridedamas naujas faktas: c.
4. Taikoma taisyklė R9: s -> d. Prie faktų pridedamas naujas faktas: d.
5. Taikoma taisyklė R11: a -> e. Prie faktų pridedamas naujas faktas: e.
6. Taikoma taisyklė R13: e -> f. Prie faktų pridedamas naujas faktas: f.
7. Taikoma taisyklė R15: f -> g. Prie faktų pridedamas naujas faktas: g.

```
8. Taikoma taisyklė R17: g -> h. Prie faktų pridedamas naujas faktas: h.
9. Taikoma taisyklė R19: h -> i. Prie faktų pridedamas naujas faktas: i.
10. Taikoma taisyklė R21: h -> j. Prie faktų pridedamas naujas faktas: j.
11. Taikoma taisyklė R23: i -> k. Prie faktų pridedamas naujas faktas: k.
12. Taikoma taisyklė R25: k -> l. Prie faktų pridedamas naujas faktas: l.
13. Taikoma taisyklė R27: l -> m. Prie faktų pridedamas naujas faktas: m.
14. Taikoma taisyklė R29: m -> n. Prie faktų pridedamas naujas faktas: n.
15. Taikoma taisyklė R31: m -> o. Prie faktų pridedamas naujas faktas: o.
16. Taikoma taisyklė R33: n -> p. Prie faktų pridedamas naujas faktas: p.
17. Taikoma taisyklė R35: o -> q. Prie faktų pridedamas naujas faktas: q.
18. Taikoma taisyklė R39: q -> r. Prie faktų pridedamas naujas faktas: r.
19. Taikoma taisyklė R43: c -> u. Prie faktų pridedamas naujas faktas: u.
20. Taikoma taisyklė R45: d -> v. Prie faktų pridedamas naujas faktas: v.
21. Taikoma taisyklė R47: v -> w. Prie faktų pridedamas naujas faktas: w.
22. Taikoma taisyklė R49: v -> x. Prie faktų pridedamas naujas faktas: x.
23. Taikoma taisyklė R51: w -> z. Prie faktų pridedamas naujas faktas: z.
24. Taikoma taisyklė R53: z -> t. Prie faktų pridedamas naujas faktas: t.
25. Taikoma taisyklė R1: t -> @. Prie faktų pridedamas naujas faktas: @.
26. Tikslas su duota produkcijų sistema pasiektas!
```

Rezultatas

```
Uždavinio sprendimas: {R3; R5; R7; R9; R11; R13; R15; R17; R19; R21; R23;
 R25; R27; R29; R31; R33; R35; R39; R43; R45; R47; R49; R51; R53; R1}.
```

Kelias labirinte



3.7. pav.: Kelias labirinte.

4. ATBULINIS IŠVEDIMAS

4.1. Pseudokodas

Answer yra taisyklių, kurios buvo pritaikytos iki pasiekiant galutinę būseną, aibė. G žymi tikslą, R - taisyklių sąrašą, F - pradiniai faktai, NF - nauji faktai, AR - pritaikytinų taisyklių aibė, TD - ankščiau bandyti gauti tikslai.

```
1: GLOBAL Answer := ();
2: GLOBAL NF := ();
3: GLOBAL TD := ();
4: function backward chaining(G, R, F)
5:     AR := ();
6:     if G nėra tarp F then
7:         while Yra pritaikytinų taisyklių su tikslu G do
8:             Pritaikytiną taisyklę pridedame į AR galą;
9:             Pritaikytiną taisyklę pažymime kaip panaudotą;
10:        end while
11:        if G yra tarp TD then
12:            Ciklas;
13:            EMPTY AR;
14:        end if
15:        for all R ∈ AR do
16:            if G nėra tarp NF then
17:                Pridedame R rezultatą į TD;
18:                for all C ∈ R sąlygos do
19:                    if C nėra tarp NF then
20:                        backward chaining(C, R, F);
21:                    end if
22:                end for
23:                if R sąlygos yra tarp NF then
24:                    Pridedame R rezultatą į NF;
25:                    Pridedame R prie Answer;
26:                end if
27:                Nustatome R kaip nepanaudotą;
28:                Pašaliname R rezultatą iš TD;
29:            end if
30:        end for
```

31: **else**
32: G pridedame į NF galą;
33: **end if**
34: **end function**

4.2. Pavyzdžiai

4.2.1. Pirmas pavyzdys - 5 produkcijų pasiekiamas tikslas

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
EB                              # R1  B -> E
FB                              # R2  B -> F
BA                              # R3  A -> B
CB                              # R4  B -> C
DC                              # R5  C -> D

2. Faktai
A

3. Tikslas
D

Failo pabaiga.
```

Duomenys

```
Produkcijos:

R1: B -> E
R2: B -> F
R3: A -> B
R4: B -> C
R5: C -> D

Duomenys:
A

Tikslas:
D
```

Programos žingsniai

```
1. Einamas tikslas: D. Rasta taisyklė R5: C -> D. Nauji tikslai: C
    2. Einamas tikslas: C. Rasta taisyklė R4: B -> C. Nauji tikslai: B
        3. Einamas tikslas: B. Rasta taisyklė R3: A -> B. Nauji tikslai: A
            4. Tikslas A yra tarp faktų.
```

Rezultatas

```
Tikslas su duota produkcijų sistema pasiektas!
Uždavinio sprendimas: {R3; R4; R5}.
```

Semantinis gafas

Semantinis grafas pateiktas 4.1. paveiklėlyje.

4.1. pav.: 5 produkcijų su pasiekiamu tikslu sekmantinis grafas.

4.2.2. Antras pavyzdys - 3 produkcijų nepasiekiamas tikslas

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
ZFB                             # R1  FB -> Z
FCD                             # R2  CD -> F
DG                              # R3  G -> D

2. Faktai
ABC

3. Tikslas
Z

Failo pabaiga.
```

Duomenys

```
Produkcijos:

R1: F, B -> Z
R2: C, D -> F
```

```
R3: G -> D

Duomenys:
A, B, C

Tikslas:
Z
```

Programos žingsniai

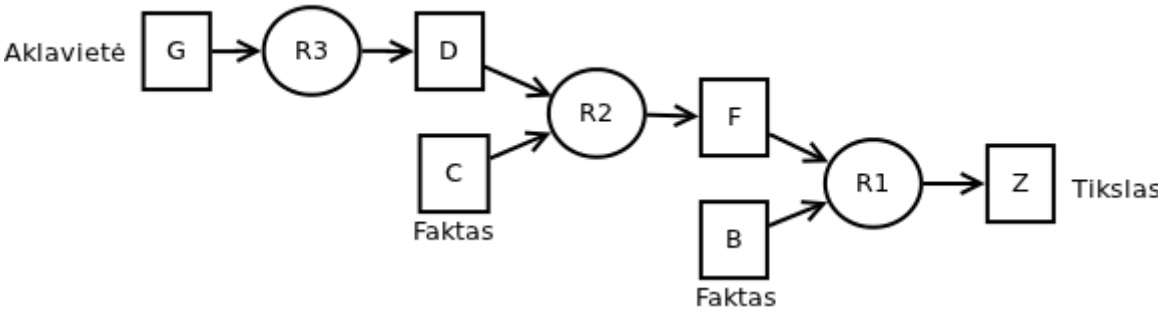
```
1. Einamas tikslas: Z. Rasta taisyklė R1: F, B -> Z. Nauji tikslai: F, B
    2. Einamas tikslas: F. Rasta taisyklė R2: C, D -> F. Nauji tikslai: C, D
        3. Tikslas C yra tarp faktų.
        3. Einamas tikslas: D. Rasta taisyklė R3: G -> D. Nauji tikslai: G
            4. Aklavietė. Nėra taisyklių šio fakto išvedimui.
    2. Tikslas B yra tarp faktų.
```

Rezultatas

```
Galutinio tikslo pasiekti neįmanoma.
Uždavinio sprendimas duotoje produkcijų sistemoje neegzistuoja.
```

Semantinis gafas

Semantinis grafas pateiktas 4.2. paveiklėlyje.



4.2. pav.: 3 produkcijų su nepasiekiamu tikslu sekmantinis grafas.

4.2.3. Trečias pavyzdys - 9 produkcijų privalomas pavyzdys

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
ZCD                             # R1  DC -> Z
DC                              # R2  C -> D
CB                              # R3  B -> C
```

```
BA                              # R4  A -> B
AD                              # R5  D -> A
DT                              # R6  T -> D
AG                              # R7  G -> A
BH                              # R8  H -> B
CJ                              # R9  J -> C

2. Faktai
T

3. Tikslas
Z

Failo pabaiga.
```

Duomenys

```
Produkcijos:

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

Duomenys:
T

Tikslas:
Z
```

Programos žingsniai

```
1. Einamas tikslas: Z. Rasta taisyklė R1: C, D -> Z. Nauji tikslai: C, D
    2. Einamas tikslas: C. Rasta taisyklė R3: B -> C. Nauji tikslai: B
        3. Einamas tikslas: B. Rasta taisyklė R4: A -> B. Nauji tikslai: A
            4. Einamas tikslas: A. Rasta taisyklė R5: D -> A. Nauji tikslai: D
                5. Einamas tikslas: D. Rasta taisyklė R2: C -> D. Nauji tikslai: C
                    6. Aklavietė. Nėra taisyklių šio fakto išvedimui.
                       Ciklas su tikslu: C ------------------------
```

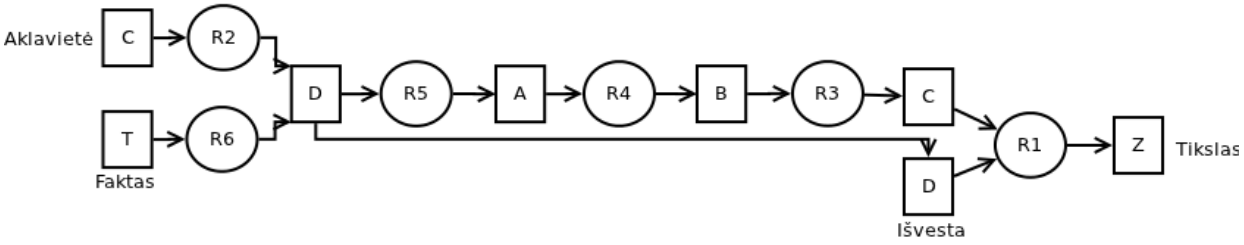
```
                5. Einamas tikslas: D. Rasta taisyklė R6: T -> D. Nauji tikslai: T
                    6. Tikslas T yra tarp faktų.
    2. Tikslas: D jau išvestas anksčiau.
```

Rezultatas

```
Tikslas su duota produkcijų sistema pasiektas!
Uždavinio sprendimas: {R6; R5; R4; R3; R1}.
```

Semantinis gafas

Semantinis grafas pateiktas 4.3. paveiklėlyje.



4.3. pav.: 9 produkcijų privalomo pavyzdžio sekmantinis grafas.

4.2.4. Ketvirtas pavyzdys - 9 produkcijų privalomas pavyzdys, sukeistos Z prielaidos

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
ZDC                             # R1  DC -> Z
DC                              # R2  C -> D
CB                              # R3  B -> C
BA                              # R4  A -> B
AD                              # R5  D -> A
DT                              # R6  T -> D
AG                              # R7  G -> A
BH                              # R8  H -> B
CJ                              # R9  J -> C

2. Faktai
T

3. Tikslas
Z

Failo pabaiga.
```

Duomenys

```
Produkcijos:

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

Duomenys:
T

Tikslas:
Z
```

Programos žingsniai

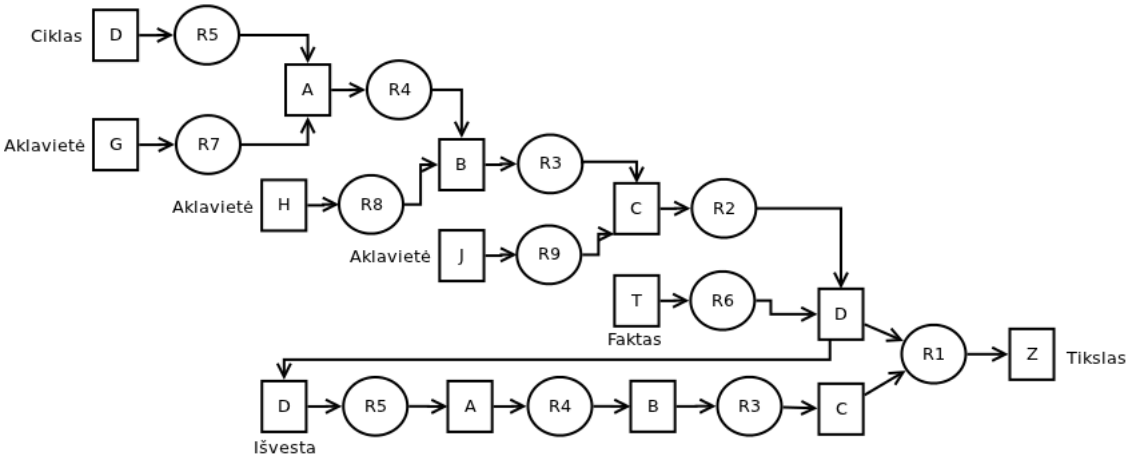
```
1. Einamas tikslas: Z. Rasta taisyklė R1: D, C -> Z. Nauji tikslai: D, C
    2. Einamas tikslas: D. Rasta taisyklė R2: C -> D. Nauji tikslai: C
        3. Einamas tikslas: C. Rasta taisyklė R3: B -> C. Nauji tikslai: B
            4. Einamas tikslas: B. Rasta taisyklė R4: A -> B. Nauji tikslai: A
                5. Einamas tikslas: A. Rasta taisyklė R5: D -> A. Nauji tikslai: D
                    6. Aklavietė. Nėra taisyklių šio fakto išvedimui.
                       Ciklas su tikslu: D ------------------------
                5. Einamas tikslas: A. Rasta taisyklė R7: G -> A. Nauji tikslai: G
                    6. Aklavietė. Nėra taisyklių šio fakto išvedimui.
            4. Einamas tikslas: B. Rasta taisyklė R8: H -> B. Nauji tikslai: H
                5. Aklavietė. Nėra taisyklių šio fakto išvedimui.
        3. Einamas tikslas: C. Rasta taisyklė R9: J -> C. Nauji tikslai: J
            4. Aklavietė. Nėra taisyklių šio fakto išvedimui.
    2. Einamas tikslas: D. Rasta taisyklė R6: T -> D. Nauji tikslai: T
        3. Tikslas T yra tarp faktų.
    2. Einamas tikslas: C. Rasta taisyklė R3: B -> C. Nauji tikslai: B
        3. Einamas tikslas: B. Rasta taisyklė R4: A -> B. Nauji tikslai: A
            4. Einamas tikslas: A. Rasta taisyklė R5: D -> A. Nauji tikslai: D
                5. Tikslas: D jau išvestas anksčiau.
```

Rezultatas

```
Tikslas su duota produkcijų sistema pasiektas!
Uždavinio sprendimas: {R6; R5; R4; R3; R1}.
```

Semantinis gafas

Semantinis grafas pateiktas 4.5. paveiklėlyje.



4.4. pav.: 9 produkcijų su sukeistomis Z prielaidomis sekmantinis grafas.

4.2.5. Penktas pavyzdys - tikslas tarp faktų

Duomenų failas

```
1  Egidijus Lukauskas
2  1. Taisyklių aibė
3  ZFB                              # R1  FB -> Z
4  FCD                              # R2  CD -> F
5  DG                               # R3  G -> D
6
7  2. Faktai
8  ABCZ
9
10 3. Tikslas
11 Z
12
13 Failo pabaiga.
```

Duomenys

```
1  Produkcijos:
2
3  R1: F, B -> Z
```

```
R2: C, D -> F
R3: G -> D

Duomenys:
A, B, C, Z

Tikslas:
Z
```

Programos žingsniai

```
1. Tikslas Z yra tarp faktų.
```

Rezultatas

```
Galutinis tikslas jau pateiktas tarp faktų.
Uždavinio sprendimas: {}.
```

Semantinis gafas

Semantinis grafas pateiktas 4.5. paveiklėlyje.

4.5. pav.: 3 produkcijų su tikslu tarp faktų sekmantinis grafas.

4.2.6. Šeštas pavyzdys - labirinto kelias nagrinėtas paskaitoje

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
@y
@t
as
sa
bs
sb
cs
sc
ds
sd
ea
ae
fe
ef
gf
fg
hg
gh
ih
hi
jh
hj
ki
ik
lk
kl
ml
lm
nm
mn
om
mo
pn
np
qo
oq
jq
qj
rq
qr
br
rb
uc
cu
vd
dv
wv
vw
xv
vx
zw
wz
tz
zt
yv
vy

2. Faktai
s

3. Tikslas
@

Failo pabaiga.
```

Įvestis

```
Produkcijos:

R1: y -> @
R2: t -> @
R3: s -> a
R4: a -> s
R5: s -> b
R6: b -> s
R7: s -> c
R8: c -> s
R9: s -> d
R10: d -> s
R11: a -> e
R12: e -> a
R13: e -> f
R14: f -> e
R15: f -> g
R16: g -> f
R17: g -> h
R18: h -> g
R19: h -> i
R20: i -> h
R21: h -> j
R22: j -> h
R23: i -> k
R24: k -> i
R25: k -> l
R26: l -> k
R27: l -> m
R28: m -> l
R29: m -> n
R30: n -> m
R31: m -> o
R32: o -> m
R33: n -> p
R34: p -> n
R35: o -> q
R36: q -> o
R37: q -> j
R38: j -> q
R39: q -> r
R40: r -> q
R41: r -> b
R42: b -> r
R43: c -> u
R44: u -> c
R45: d -> v
R46: v -> d
R47: v -> w
R48: w -> v
R49: v -> x
R50: x -> v
R51: w -> z
R52: z -> w
R53: z -> t
R54: t -> z
R55: v -> y
R56: y -> v

Duomenys:
s

Tikslas:
@
```

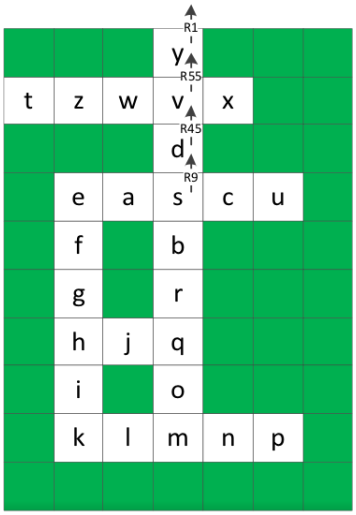
Programos žingsniai

1. Einamas tikslas: @. Rasta taisyklė R1: y -> @. Nauji tikslai: y
    2. Einamas tikslas: y. Rasta taisyklė R55: v -> y. Nauji tikslai: v
        3. Einamas tikslas: v. Rasta taisyklė R45: d -> v. Nauji tikslai: d
            4. Einamas tikslas: d. Rasta taisyklė R9: s -> d. Nauji tikslai: s
                5. Tikslas s yra tarp faktų.

Rezultatas

```
Tikslas su duota produkcijų sistema pasiektas!
Uždavinio sprendimas: {R9; R45; R55; R1}.
```

Kelias labirinte



4.6. pav.: Kelias labirinte per y.

4.2.7. Septintas pavyzdys - labirinto kelias nagrinėtas paskaitoje, kitas išėjimas

Duomenų failas

```
Egidijus Lukauskas
1. Taisyklių aibė
@t
@y
as
sa
bs
sb
cs
sc
ds
sd
ea
ae
fe
ef
gf
fg
hg
gh
ih
hi
jh
hj
ki
ik
lk
kl
ml
lm
nm
mn
om
mo
pn
np
qo
oq
jq
qj
rq
qr
br
rb
uc
cu
vd
dv
wv
vw
xv
vx
zw
wz
tz
zt
yv
vy

2. Faktai
s

3. Tikslas
@

Failo pabaiga.
```

Įvestis

```
Produkcijos:

R1: t -> @
R2: y -> @
R3: s -> a
R4: a -> s
R5: s -> b
R6: b -> s
R7: s -> c
R8: c -> s
R9: s -> d
R10: d -> s
R11: a -> e
R12: e -> a
R13: e -> f
R14: f -> e
R15: f -> g
R16: g -> f
R17: g -> h
R18: h -> g
R19: h -> i
R20: i -> h
R21: h -> j
R22: j -> h
R23: i -> k
R24: k -> i
R25: k -> l
R26: l -> k
R27: l -> m
R28: m -> l
R29: m -> n
R30: n -> m
R31: m -> o
R32: o -> m
R33: n -> p
R34: p -> n
R35: o -> q
R36: q -> o
R37: q -> j
R38: j -> q
R39: q -> r
R40: r -> q
R41: r -> b
R42: b -> r
R43: c -> u
R44: u -> c
R45: d -> v
R46: v -> d
R47: v -> w
R48: w -> v
R49: v -> x
R50: x -> v
R51: w -> z
R52: z -> w
R53: z -> t
R54: t -> z
R55: v -> y
R56: y -> v

Duomenys:
s

Tikslas:
@
```

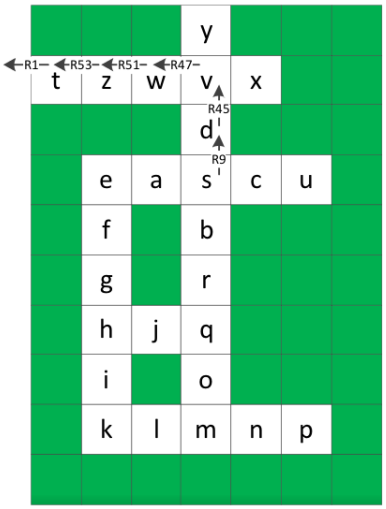
Programos žingsniai

```
1. Einamas tikslas: @. Rasta taisyklė R1: t -> @. Nauji tikslai: t
   2. Einamas tikslas: t. Rasta taisyklė R53: z -> t. Nauji tikslai: z
      3. Einamas tikslas: z. Rasta taisyklė R51: w -> z. Nauji tikslai: w
         4. Einamas tikslas: w. Rasta taisyklė R47: v -> w. Nauji tikslai: v
            5. Einamas tikslas: v. Rasta taisyklė R45: d -> v. Nauji tikslai: d
               6. Einamas tikslas: d. Rasta taisyklė R9: s -> d. Nauji tikslai: s
                  7. Tikslas s yra tarp faktų.
```

Rezultatas

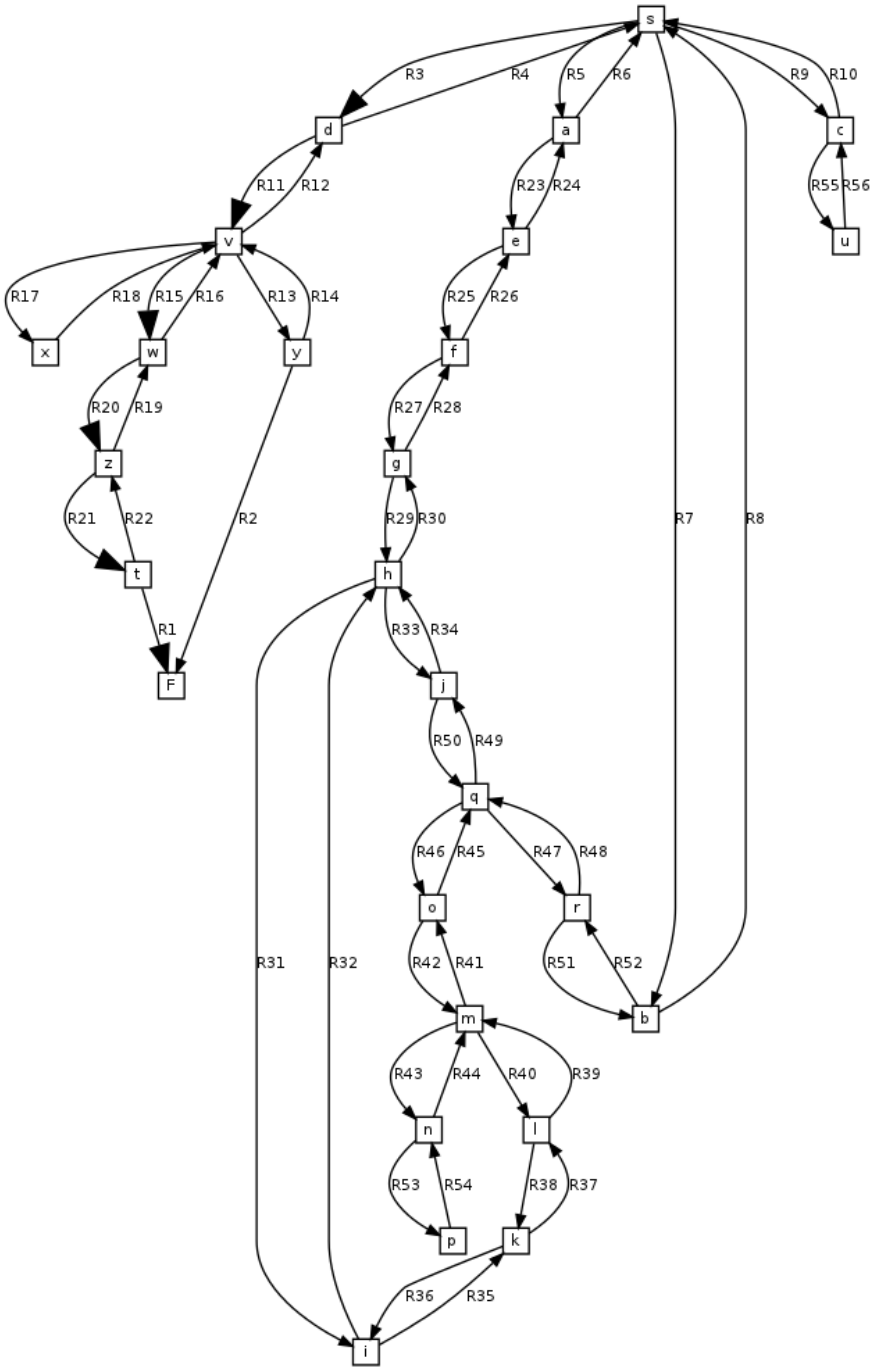
```
Tikslas su duota produkcijų sistema pasiektas!
Uždavinio sprendimas: {R9; R45; R47; R51; R53; R1}.
```

Kelias labirinte



4.7. pav.: Kelias labirinte per t.

Labirinto kelių grafas



4.8. pav.: labirinto kelių grafas.