

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Programa
„Atbulinis išvedimas ruby kalba“

Darbą atliko: 3 Kurso 1 grupės studentas
Rytis Karpuška

Vilnius
2014

Turiny

Turiny	2
1. Darbo tikslas	3
2. Atbulinio išvedimo algoritmas	3
2.1 Detalus aprašymas	3
2.1.1 Įeitis	3
2.1.2 Išeitis	3
2.1.3 Algoritmo veikimas	3
2.2 Struktūrinė schema	4
2.3 Pseudokodas	5
2.4 Klasų diagrama	6
2.5 Ruby Kodas	6
2.6 Pavyzdžiai	12
2.6.1 Pavyzdys 1	12
2.6.2 Pavyzdys 2	13
2.6.3 Pavyzdys 3	15
2.6.4 Pavyzdys 4	16
2.6.5 Pavyzdys 5	17
2.6.6 Pavyzdys 6	18
2.6.7 Pavyzdys 7	19
2.6.8 Pavyzdys 8	20
2.6.9 Pavyzdys 9	21
2.6.10 Pavyzdys 10	22
2.6.11 Pavyzdys 11	24

1. Darbo tikslas

Šio darbo tikslas yra įgyvendinti ir aprašyti atbulinio išvedimo algoritmą.

2. Atbulinio išvedimo algoritmas

Atbulinio išvedimo algoritmas yra vienas iš dviejų pagrindinių protavimo algoritmų naudojamų išvedimo varikliuose.

2.1 Detalus aprašymas

2.1.1 Įeitis

Algoritmas turi tokią įeitį

- Taisyklės (produkcijos), išreikštos loginės implikacijos forma
- Faktai
- Tikslas

2.1.2 Išėjis

Algoritmas turi tokią išeitį:

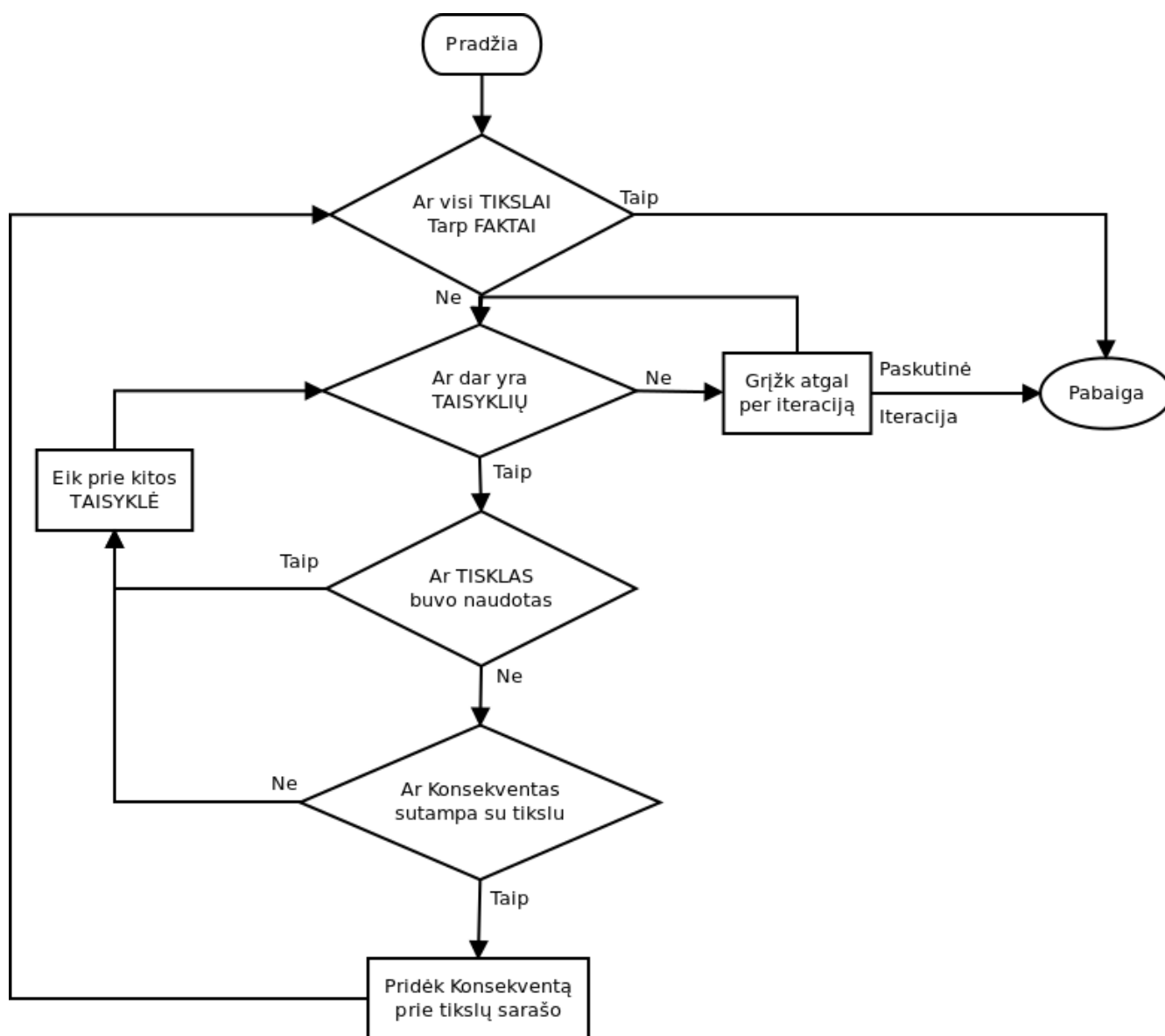
- Vėliavėlę ar tikslas buvo sėkmingai pasiektas
- Kelias iki tikslo (išreikštas pritaikytomis taisyklėmis)

2.1.3 Algoritmo veikimas

Atbulinio išvedimo algoritmas pradeda darbą turėdamas sąrašą tikslų (pradinėse sąlygose jis yra tik vienas) ieško tokių taisyklių, kurių antecedentai būtų žinomi kaip tiesa (būtų tarp faktų). Jeigu antecedentas nėra žinomas kaip tiesa, jis pridedamas prie tikslų aibės ir pradedama nauja iteracija. Bendrai algoritmas susideda iš tokių žingsnių:

1. Patikriname ar dabartinis tikslas yra tarp faktų, jeigu taip, grįžtame prie senesės iteracijos
(Kode bus pažymėta MARK1)
2. Patikriname ar dabartinis tikslas nebuvo jau bandomas išvesti, t.y. Tikriname ar nepatekome į ciklą
(kode bus pažymėta MARK2)
3. Surandame taisyklę, kurios konsekvantas yra mūsų tikslas
(kode bus pažymėta MARK3)
4. Surastos taisyklės antecendai pridedami prie taisyklių sąrašo ir pradedama nauja iteracija.
(kode bus pažymėta MARK4)

2.2 Struktūrinė schema



2.3 Pseudokodas

Pateikiamas rekursinis algoritmo variantas

```
LAIKINAS_KELIAS := SUKURTI NAUJĄ MASYVĄ
KELIAS := SUKURTI NAUJĄ MASYVĄ
PROCEDURE IŠSPRĘSK(TAISYKLĖS, FAKTAI, TIKSLAS)
    IF TIKSLAS YRA TARP FAKTAI THEN                                     <<MARK1
        LAIKINAS_KELIAS.ISIMK(PASKUTINIS)
        RETURN RASTA
    ENDIF

    IF TIKSLAS IN LAIKINAS_KELIAS                                       <<MARK2
        LAIKINAS_KELIAS.ISIMK(PASKUTINIS)
        RETURN FALSE
    ENDIF

    FOREACH TAISYKLĖ IN TAISYKLĖS DO
        IF NOT TAISYKLĖ.REZULTATAS = TIKSLAS THEN                       <<MARK3
            CONTINUE
        ENDIF

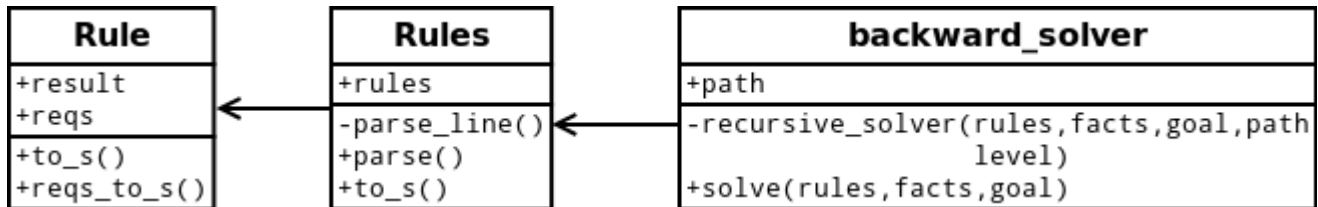
        VISI_YRA := TRUE
        FOREACH ANTECEDENTAS IN TAISYKLĖ DO
            LAIKINAS_KELIAS.PRIDEK(TIKSLAS)
            R := IŠSPRĘSK(TAISYKLĖS, FAKTAI, ANTECEDENTAS)               <<MARK4
            IF R = FALSE THEN
                VISI_YRA := FALSE
                LAIKINAS_KELIAS.ISIMK(PASKUTINIS)
                KELIAS.ISVALYK
            ENDIF

            IF VISI_YRA = TRUE THEN
                KELIAS.PRIDEK(TAISYKLĖ)
            ENDIF
        ENDFOR

    ENDFOR

    RETURN FALSE
ENDPROC
```

2.4 Klasių diagrama



2.5 Ruby Kodas

backward_solver.rb

```
load 'chaining_rules.rb'
```

```
class Backward_solver
```

```
  @resultText
```

```
  @path
```

```
  @tmpFacts
```

```
  attr_accessor :resultText
```

```
  attr_accessor :path
```

```
  attr_accessor :tmpFacts
```

```
  def initialize()
```

```
    @resultText = Array.new
```

```
    @path = Array.new
```

```
    @tmpFacts = Array.new
```

```
  end
```

```
  def recursive_solver(rules, facts, goal, path, level)
```

```
    @resultText << Array.new
```

```
    @tmpFacts << Array.new
```

```
    @resultText[-1] << level
```

```
    @resultText[-1] << 'Goal ' + goal + ", "
```

```
    if(facts.include? Goal)
```

```
      @resultText[-1][-1] << "Goal is between facts"
```

```
      path.delete_at(path.length - 1)
```

```
      return true
```

```
    end
```

```
    if(path.include? Goal)
```

```
      @resultText[-1][-1] << "Loop detected"
```

```
      path.delete_at(path.length - 1)
```

```
      return false
```

```
    end
```

<<MARK1

<<MARK2

```

#search for rule with goal
rules.each_with_index do |rule, i|
  if(rule.result != goal)
    next
  end

  @resultText[-1][-1] << 'Using rule ' + rule.to_s + ', '
  @resultText[-1][-1] << 'New goals ' + rule.reqs_to_s

  #check for every requirement
  allReqsFullfiled = true
  rule.req.each do |req|
    path << goal
    result = recursive_solver(rules, facts, req, path, level + 1) <<MARK4
    if(result == false)
      @resultText << Array.new
      @resultText[-1] << level
      @resultText[-1] << 'Goal ' + goal + ", "
      allReqsFullfiled = false
      path.delete_at(path.length - 1)
      @path.clear
      break
    end
  end

  #check if we reached goal
  if(allReqsFullfiled)
    @path << i;
    return true
  end
end

@resultText[-1][-1] << ' No suitable rule found for current goal'
path.delete_at(path.length - 1)
return false
end

def solve(rules, facts, goal)
  #initialize
  @resultText.clear
  @tmpFacts.clear
  @path.clear

  path = Array.new

  #solve recursively
  result = recursive_solver(rules, facts, goal, path, 1)

```

```
        return result
```

```
    end
```

```
end
```


chaining_rules.rb

```
load 'chaining_rule.rb'
```

```
class ChainingRules
```

```
  @rules
```

```
  @errorLine
```

```
  @errorString
```

```
  attr_accessor :rules, :errorLine, :errorString
```

```
  def initialize()
```

```
    @rules = []
```

```
    @errorLine = 0
```

```
    @errorString = ""
```

```
  end
```

```
  def parse_line(line, index)
```

```
    r = ChainingRule.new
```

```
    words = line.split
```

```
    if(words.size == 0)
```

```
      return :ok
```

```
    end
```

```
    first_word = true
```

```
    acceptable = false
```

```
    words.each do |word|
```

```
      if(word.start_with?("//"))
```

```
        break
```

```
      end
```

```
      #check if we have comment without spaces
```

```
      if(word.include?("//"))
```

```
        commentStarted = true
```

```
        word = word.split("//").first
```

```
      end
```

```
      if(word.length > 1)
```

```
        @errorLine = index
```

```
        @errorString = "Rule argument larger than 1 symbol"
```

```
        return :parseError
```

```
      end
```

```
      if(first_word)
```

```
        r.result = word
```

```
        first_word = false
```

```
      else
```

```
        r.req << word
```

```
        acceptable = true
```

```

        end

        if commentStarted
            break
        end
    end

    end

    if(acceptable)
        @rules << r

    elsif(!acceptable && first_word == false)
        @errorLine = index
        @errorString = "Not enough data to form a rule"
        return :parseError
    end

    return :ok
end

def parse(text)
    text.lines.each_with_index do |line, index|
        res = parse_line(line, index + 1)
        if(res != :ok)
            return res
        end
    end
    return :ok
end

def to_s()
    str = ""
    rules.each do |rule|
        str += rule.to_s() + "\n"
    end
    return str
end
end
end

```

chaining_rule.rb

```
class ChainingRule
  @req
  @result
  @used
  attr_accessor :req
  attr_accessor :result
  attr_accessor :used

  def initialize()
    @req = Array.new
    @result = ""
    @used = false
  end

  def to_s()
    str = ""
    @req.each_with_index do |req, index|
      str += req
      if(index < @req.size - 1)
        str += ", "
      end
    end

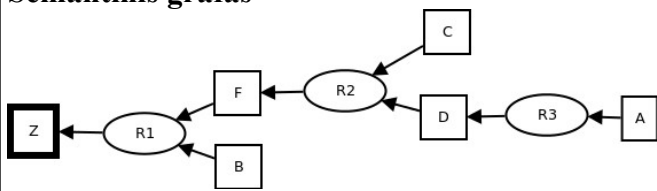
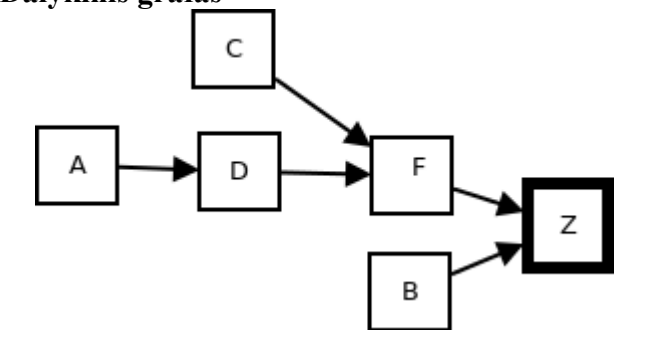
    str += " -> " + @result
    return str
  end

  def reqs_to_s()
    str = ""
    @req.each do |req|
      str += req + " "
    end

    return str
  end
end
```

2.6 Pavyzdžiai

2.6.1 Pavyzdys 1

<p>Semantinis grafas</p> 	<p>Dalykinis grafas</p> 
<p>Ieitis</p> <p>Z F B F C D D A</p> <p>A B C</p> <p>Z</p>	
<p>Eiga</p> <p>Algorithm Trace:</p> <p>---->Goal Z, Using rule F, B -> Z, New goals F B</p> <p>---->---->Goal F, Using rule C, D -> F, New goals C D</p> <p>---->---->---->Goal C, Goal is between facts</p> <p>---->---->---->Goal D, Using rule A -> D, New goals A</p> <p>---->---->---->---->Goal A, Goal is between facts</p> <p>---->---->---->---->Goal B, Goal is between facts</p>	
<p>Išeitis</p> <p>Goal reached successfully</p> <p>Path: R3, R2, R1</p>	

2.6.2 Pavyzdys 2

<p>Semantinis grafas</p> <pre> graph LR Z[Z] D1[D] C[C] B[B] T[T] A[A] R1((R1)) R3((R3)) R5((R5)) R4((R4)) R6((R6)) D1 --> R1 C --> R1 R1 --> Z B --> R3 R3 --> C D2[D] --> R5 R5 --> A A --> R4 R4 --> B T --> R6 R6 --> D2 </pre>	<p>Dalykinis grafas</p> <pre> graph LR G[G] --> A[A] A --> B[B] B --> C[C] C --> Z[Z] D1[D] --> A T[T] --> D1 H[H] --> B J[J] --> C D2[D] --> Z </pre>
<p>Ieitis</p> <p>Z D C D C C B B A A D D T A G B H C J</p> <p>T</p> <p>Z</p>	
<p>Eiga</p> <p>Algorithm Trace:</p> <p>---->Goal Z, Using rule D, C -> Z, New goals D C</p> <p>---->---->Goal D, Using rule C -> D, New goals C</p> <p>---->---->---->Goal C, Using rule B -> C, New goals B</p> <p>---->---->---->---->Goal B, Using rule A -> B, New goals A</p> <p>---->---->---->---->---->Goal A, Using rule D -> A, New goals D</p> <p>---->---->---->---->---->---->Goal D, Loop detected</p> <p>---->---->---->---->---->---->Goal A, Using rule G -> A, New goals G</p> <p>---->---->---->---->---->---->---->Goal G, No suitable rule found for current goal</p> <p>---->---->---->---->---->---->---->Goal A, No suitable rule found for current goal</p> <p>---->---->---->---->---->---->---->Goal B, Using rule H -> B, New goals H</p>	


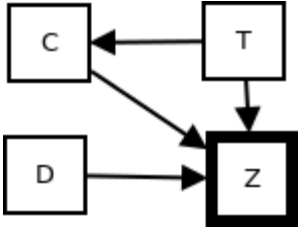
---->---->---->---->---->Goal H, No suitable rule found for current goal
---->---->---->---->Goal B, No suitable rule found for current goal
---->---->---->Goal C, Using rule J -> C, New goals J
---->---->---->---->Goal J, No suitable rule found for current goal
---->---->---->Goal C, No suitable rule found for current goal
---->---->Goal D, Using rule T -> D, New goals T
---->---->---->Goal T, Goal is between facts
---->---->Goal C, Using rule B -> C, New goals B
---->---->---->Goal B, Using rule A -> B, New goals A
---->---->---->---->Goal A, Using rule D -> A, New goals D
---->---->---->---->---->Goal D, Using rule C -> D, New goals C
---->---->---->---->---->---->Goal C, Loop detected
---->---->---->---->---->---->Goal D, Using rule T -> D, New goals T
---->---->---->---->---->---->---->Goal T, Goal is between facts

Išėitis


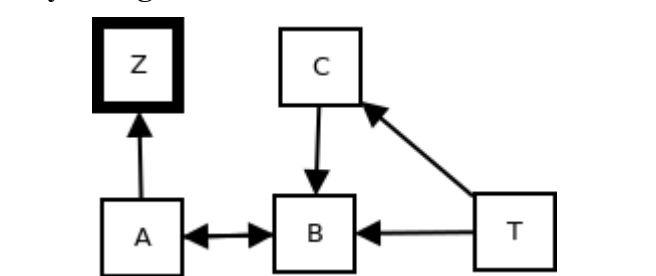
Goal reached successfully

Path: R6, R5, R4, R3, R1

2.6.3 Pavyzdys 3

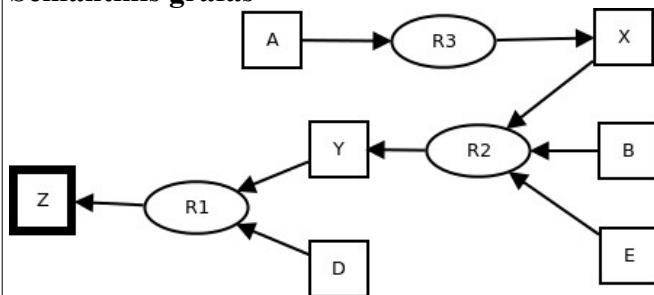
<p>Semantinis grafas</p> 	<p>Dalykinis grafas</p> 
<p>Ieitis</p> <p>Z C D C T Z T</p> <p>T</p> <p>Z</p>	
<p>Eiga</p> <p>Algorithm Trace:</p> <p>---->Goal Z, Using rule C, D -> Z, New goals C D</p> <p>---->---->Goal C, Using rule T -> C, New goals T</p> <p>---->---->---->Goal T, Goal is between facts</p> <p>---->---->Goal D, No suitable rule found for current goal</p> <p>---->Goal Z, Using rule T -> Z, New goals T</p> <p>---->---->Goal T, Goal is between facts</p>	
<p>Išėitis</p> <p>Goal reached successfully</p> <p>Path: R3</p>	

2.6.4 Pavyzdys 4

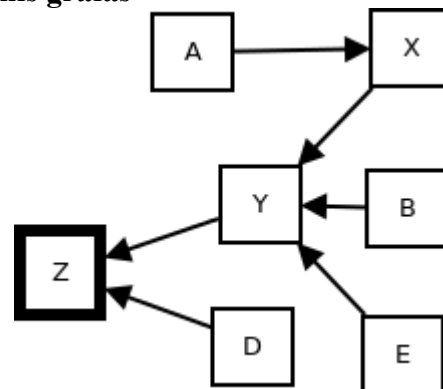
Semantinis grafas 	Dalykinis grafas 
Ieitis Z A A B B A C B T C T T Z	
Eiga Algorithm Trace: ---->Goal Z, Using rule A -> Z, New goals A ---->---->Goal A, Using rule B -> A, New goals B ---->---->---->Goal B, Using rule A, C -> B, New goals A C ---->---->---->---->Goal A, Loop detected ---->---->---->Goal B, Using rule T -> B, New goals T ---->---->---->---->Goal T, Goal is between facts	
Išėitis Goal reached successfully Path: R4, R2, R1	

2.6.5 Pavyzdys 5

Semantinis grafas



Dalykinis grafas



Ieitis

Z Y D
Y X B E
X A

A B C D E

Z

Eiga

Algorithm Trace:


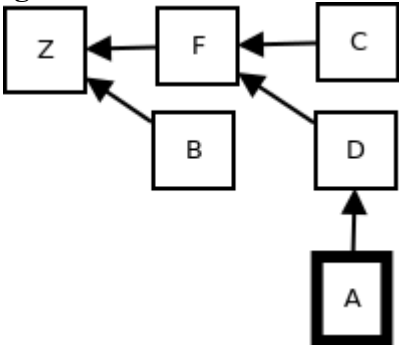
---->Goal Z, Using rule Y, D -> Z, New goals Y D
 ---->---->Goal Y, Using rule X, B, E -> Y, New goals X B E
 ---->---->---->Goal X, Using rule A -> X, New goals A
 ---->---->---->---->Goal A, Goal is between facts
 ---->---->---->Goal B, Goal is between facts
 ---->---->---->Goal E, Goal is between facts
 ---->---->Goal D, Goal is between facts

Išeitis

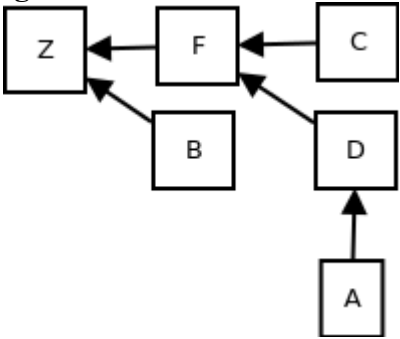
Goal reached successfully

Path: R3, R2, R1

2.6.6 Pavyzdys 6

<p>Semantinis grafas</p> 	<p>Dalykinis grafas</p>  <pre>graph RL; C --> F; F --> Z; F --> B; C --> D; B --> D; D --> A; style A stroke-width:4px;</pre>
<p>Ieitis</p> <p>Z F B F C D D A</p> <p>A</p> <p>A</p>	
<p>Eiga</p>	
<p>Išeitis</p> <p>Goal reached successfully</p>	

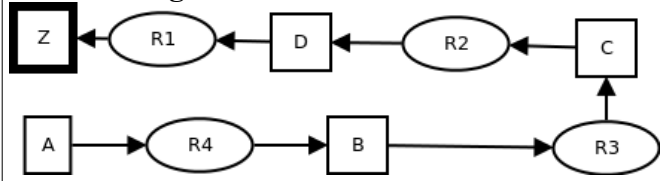
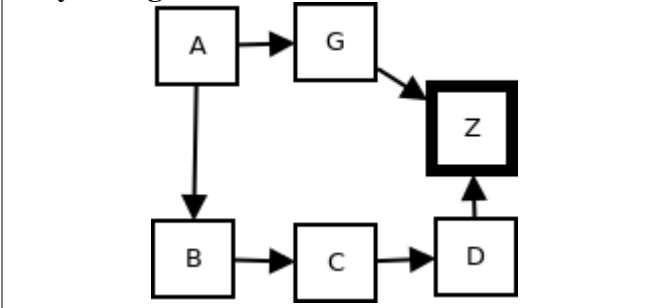
2.6.7 Pavyzdys 7

Semantinis grafas	Dalykinis grafas
	
kĳeitis	
Z F B	
F C D	
D A	
A B C	
W	
Eiga	
Iĳeitis	
Goal Could not be reached	

2.6.8 Pavyzdys 8

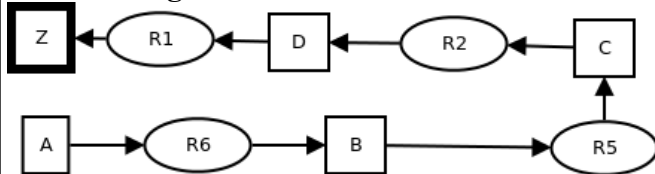
<p>Semantinis grafas</p> <pre>graph LR; Z[Z] --> R1((R1)); R1 --> G[G]; G --> R2((R2)); R2 --> A[A];</pre>	<p>Dalykinis grafas</p> <pre>graph TD; A[A] --> G[G]; G --> Z[Z]; A --> B[B]; B --> C[C]; C --> D[D]; D --> Z;</pre>
<p>Ieitis</p> <p>Z G G A B A C B D C Z D</p> <p>A</p> <p>Z</p>	
<p>Eiga</p> <p>Algorithm Trace:</p> <p>---->Goal Z, Using rule $G \rightarrow Z$, New goals G</p> <p>---->---->Goal G, Using rule $A \rightarrow G$, New goals A</p> <p>---->---->---->Goal A, Goal is between facts</p>	
<p>Išeitis</p> <p>Goal reached successfully</p> <p>Path: R2, R1</p>	

2.6.9 Pavyzdys 9

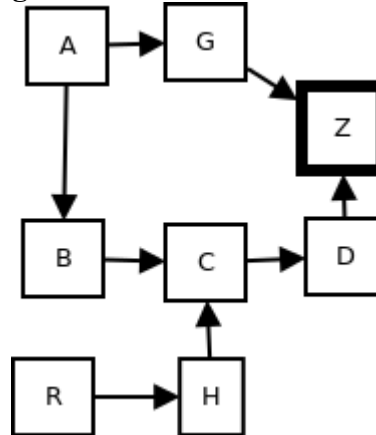
<p>Semantinis grafas</p>  <pre> graph LR Z[Z] --> R1((R1)) R1 --> D[D] D --> R2((R2)) R2 --> C[C] C --> R3((R3)) R3 --> B[B] B --> R4((R4)) R4 --> A[A] </pre>	<p>Dalykinis grafas</p>  <pre> graph TD A[A] --> G[G] G --> Z[Z] A --> B[B] B --> C[C] C --> D[D] D --> Z </pre>
<p>Ieitis</p> <p>Z D D C C B B A G A Z G</p> <p>A</p> <p>Z</p>	
<p>Eiga</p> <p>Algorithm Trace:</p> <p>---->Goal Z, Using rule D -> Z, New goals D</p> <p>---->---->Goal D, Using rule C -> D, New goals C</p> <p>---->---->---->Goal C, Using rule B -> C, New goals B</p> <p>---->---->---->---->Goal B, Using rule A -> B, New goals A</p> <p>---->---->---->---->---->Goal A, Goal is between facts</p>	
<p>Išeitis</p> <p>Goal reached successfully</p> <p>Path: R4, R3, R2, R1</p>	

2.6.10 Pavyzdys 10

Semantinis grafas



Dalykinis grafas



Ieitis

Z D
D C
C H
H R
C B
B A
G A
Z G

A

Z

Eiga

Algorithm Trace:

---->Goal Z, Using rule D -> Z, New goals D
 ---->---->Goal D, Using rule C -> D, New goals C
 ---->---->---->Goal C, Using rule H -> C, New goals H
 ---->---->---->---->Goal H, Using rule R -> H, New goals R
 ---->---->---->---->---->Goal R, No suitable rule found for current goal
 ---->---->---->---->Goal H, No suitable rule found for current goal
 ---->---->---->Goal C, Using rule B -> C, New goals B
 ---->---->---->---->Goal B, Using rule A -> B, New goals A
 ---->---->---->---->---->Goal A, Goal is between facts

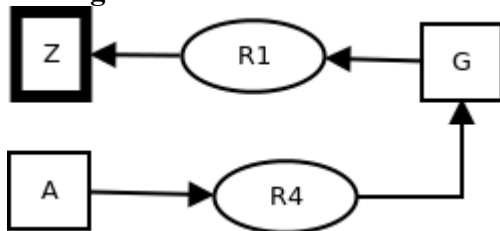
Išeitis

Goal reached successfully

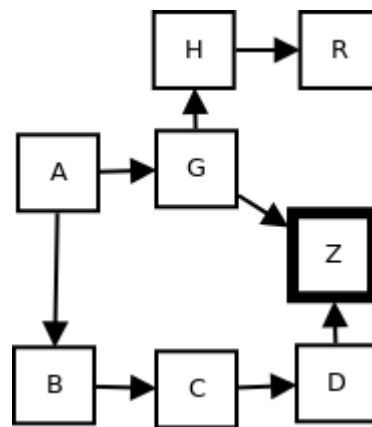
Path: R6, R5, R2, R1

2.6.11 Pavyzdys 11

Semantinis grafas



Dalykinis grafas



Ieitis

Z G
H G
G R
G A
B A
C B
D C
Z D

A

Z

Eiga

Algorithm Trace:

---->Goal Z, Using rule $G \rightarrow Z$, New goals G
 ---->---->Goal G, Using rule $R \rightarrow G$, New goals R
 ---->---->---->Goal R, No suitable rule found for current goal
 ---->---->Goal G, Using rule $A \rightarrow G$, New goals A
 ---->---->---->Goal A, Goal is between facts

Išėitis

Goal reached successfully

Path: R4, R1