Vilniaus universitetas
Matematikos ir informatikos fakultetas
Programų sistemų katedra

Atbulinio išvedimo produkcijų sistemoje demonstracinė sistema. Java kalba

Autorius: Dainius Jocas

Vilnius
2011

Turinys

Įvadas

Šis dokumentas yra skirtas dokumentuoti ir apibendrinti antrąjį kurso „Dirbtinis inteletas“ laboratorinį darbą. Laboratorinio darbo užduotis yra suprogramuoti atbulinį išvedimą produkcijų sistemoje. Mano pasirinkta programvimo kalba yra Java. Šiame dokumente pateiksiu išsamų darbo aprašymą, atbulinio išvedimo pseudokodą, programos kodą, kuris yra relevantinis algoritmui, susiesiu pseudokodą ir programos kodą. Taip pat pateiksiu tris atbulio išvedimo pavyzdžius, pademonstruosiu jų išvedimo protokolą, kurį sugeneruoja demonstracinė sistema, nupiešiu semantinius išvedimo grafus.

Žymėjimai ir paaiškinimai

1. H – atbulinio išvedimo tikslas.
2. Faktai - pradiniai teisingi duomenys.
3. Implikacija – loginys reiškinys, turintis formą „A, B -> C“. Produkcijos atitikmuo šio laboratorinio darbo kontekste.
4. Sukcedentas – dešinioji implikacijos pusė.
5. Antecedentas – keirioji implikacijos pusė.
6. Antecedento narys – loginis kintamasis esantis antecedente.

Pseudokodas

Pseudokodas	Žingsnio nr.
`atbulinis_išvedimas(H)`	{1}
`if H yra_tarp_faktų then return true`	{2}
`if H nėra_tarp_sukcedentų then return false`	{3}
`if atbulinis_isvedimas_pakliuvo_į_ciklą then return false`	{4}
`for kiekvienai_implikacijai_kur_H_yra_sukcedentas do`	{5}
`    if visiems_antecedentams`	{6}
`       true == atbulinis_išvedimas(antecedento_narys)`	{7}
`then return true`	
`return false`	{8}

Pseudokodo žodinis paaiškinimas

Atbulinis išvedimas yra rekursyvi procedūra. Visų pirma proceūra patikrina ar jos tikslas nėra tarp faktų, jei yra tada grąžinamas sėkmingos baigties požymis – true. Jei nebuvo sėkmingos baigties tikrinama ar prasminga toliau ieškoti išvedimo – tikrinama ar tikslas yra tarp sukcedentų, jei nėra tada grąžinamas nesėkmės požymis. Jei tikslas yra tarp sukcedentų tada tikrinama ar išvedimas nepakliuvo į ciklą. Jei pakliuvo, tai grąžinamas nesėkmės požymis. Jei nedirbama cikle, tai procedūra bando išvesti tikslą rekusyviais kvietimais mėgindama įrodyti kiekvieną antecedentą, kurio sukcedentas ir yra tikslas. Jei nepavyko įrodyti ir visi galimi variantai jau perrinkti, tada grąžinamas nesėkmės požymis.

Duomenų failo reikalavimai

1. Pirma failo eilutė gali būti arba tuščia arba su komentaru, kuris prasideda simboliu `#'.
2. Pradedant antra eilute privalo būti implikacijų sąrašas, kiekvieną implikaciją įrašant naujoje eilutėje. Implikacijos pavydys: ‚ABC' programa supras kaip „A, B -> C".
3. Po implikacijų sąrašo turi būti tuščia eilutė, o tada faktų sąrašas.
4. Faktai programos darbui. Pavyzdys: „ABC" programa supras kaip tris faktus „A, B, C".
5. Tuščia eilutė.
6. Atbulinio išvedimo tikslas.
7. Tuščia eilutė.

Duomenų failo pavyzdys

Eilutės numeris	Duomenų failo tekstas
1:	#Dainius Jocas
2:	FBZ
3:	CDF
4:	AD
5:	
6:	ABC
7:	
8:	Z
9:	

Relevantinis programos kodas

```
/**
 * This class is implementation of the Backward-Chaining
 * @author Dainius Jocas
 */
public class BackwardChaining {
    private String facts;
    private ArrayList<Implication> implications;
    private String goal;
    private ArrayList<String> productions = new ArrayList();
    private String path = "";

    /**
     * Constructor
     * @param goal of the chaining
     * @param facts facts for the chaining
     * @param implications which will be used for derivation of the
new facts
     */
    public BackwardChaining(String goal, String facts,
ArrayList<String> implications) {
        this.goal = goal;
        this.facts = facts;
        this.implications = new ArrayList();
        for (String implication : implications) {
            this.implications.add(new Implication(implication));
        }
    }

    /**
     * Method that checks if goal is derivable in production system.
     * @param goal
     * @return return true if goal is derivable, otherwise - false
     */
/* 1 Funkcijos deklaracija */
    public boolean doBackwardChaining(String goal) {
        System.out.println("Searching for: " + goal);
/* 2 if H yra_tarp_faktų then return true */
        if (isGoalInFacts(goal)) {
            System.out.println(goal + " is in facts.");
            return true;
        }
/* 3 if H nėra_tarp_sukcedentų then return false */
        if (!isGoalSomewhereAsConsequent(goal)) {
            System.out.println("DEADEND!");
            removeLastStep();
            return false;
        }
/* 4 if atbulinis_isvedimas_pakliuvo_į_ciklą then return false */
        if (isChainingInTheLoop(goal)) {
            System.out.println("LOOP!!!");
            return false;
        }
        boolean result = true;
/* 5 for kiekvienai_implikacijai_kur_H_yra_sukcedentas do */
        for (Implication implication : implications) {
            if (implication.getConsequent().contains(goal)) {
                addNewGoal(goal);
                printDerivationPath();
```

```
/* 6 if visiems_antecedentams */
                for (String newGoal :
implication.getAntecedentAsList()) {
/* 7 true == atbulinis_išvedimas(antecedent_narys)  then return true
*/
                    result = result && doBackwardChaining(newGoal);
                }
                if (result) {
                    removeLastStep();
                    System.out.println(goal + " goes to list of
facts!");
                    addNewFact(goal);
                    addImplicationToProductionSystem(implication);
                    return true;
                } else {
                    result = true;
                }
            }
        }
/* 8 return false*/
        return false;
    }

    /**
     * Method that checks if the goal is in facts
     * @param goal that is being searched
     * @return true if goal is in facts, otherwise - false
     */
    public boolean isGoalInFacts(String goal) {
        if (this.facts.contains(goal)) {
            return true;
        } else {
            return false;
        }
    }

    /**
     * Method that checks if there is a implications which has such a
consequent
     * @param consequent
     * @return true if there is such an implication with a specific
consequent,
     * otherwise - false
     */
    boolean isGoalSomewhereAsConsequent(String consequent) {
        for (Implication implication : this.implications) {
            if (implication.getConsequent().contains(consequent)) {
                return true;
            }
        }
        return false;
    }

    /**
     * Method that ads a fact to the list of facts
     * @param fact
     */
    private void addNewFact(String fact) {
        this.facts = this.facts + fact;
    }
```

```
    /**
     * Method that add implication to the production system
     * @param implication
     */
    void addImplicationToProductionSystem(Implication implication) {
        this.productions.add(implication.getDescriptor());
    }

    /**
     * Method that checks if chaining is in loop
     * @param goal
     * @return true if chaining is in loop, otherwise - false.
     */
    boolean isChainingInTheLoop(String goal) {
        return this.path.contains(goal);
    }
}
```

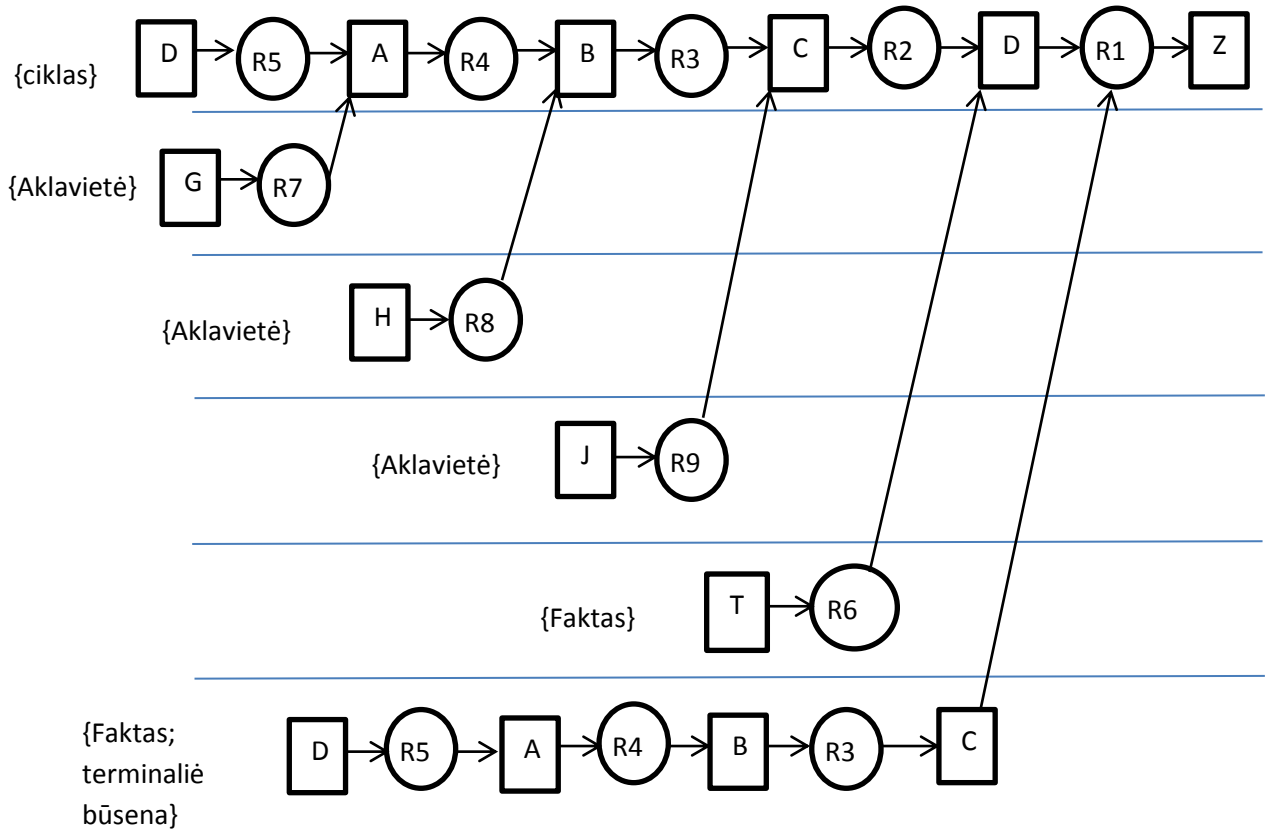
Pavyzdys: devynios implikacijos

Šiuo pavyzdžiu norima parodyti, kad atbulinio išvedimo procedūra sugeba rasti aklavietę, ciklą ir kad algoritmo žingsnių seka atitinka reikalavimus.

Duomenų failo turinys	Programos darbo protokolas
#Dainius Jocas DCZ CD BC AB DA TD GA HB JC T Z	List of implications that was read from the data file: 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 Initial list of facts: {T} The goal for the program: Z Searching for: Z Derivation path: <- Z Searching for: D Derivation path: <- D <- Z Searching for: C Derivation path: <- C <- D <- Z Searching for: B Derivation path: <- B <- C <- D <- Z Searching for: A Derivation path: <- A <- B <- C <- D <- Z Searching for: D LOOP!!! Derivation path: <- A <- B <- C <- D <- Z Searching for: G DEADEND! Derivation path: <- B <- C <- D <- Z Searching for: H DEADEND! Derivation path: <- C <- D <- Z Searching for: J DEADEND!

```
Derivation path:  <- D <- Z
Searching for: T
T is in facts.
D goes to list of facts!
Searching for: C
Derivation path:  <- C <- Z
Searching for: B
Derivation path:  <- B <- C <- Z
Searching for: A
Derivation path:  <- A <- B <- C <- Z
Searching for: D
D is in facts.
A goes to list of facts!
B goes to list of facts!
C goes to list of facts!
Z goes to list of facts!

The goal is reached!
The answer is: {R6, R5, R4, R3, R1}
```



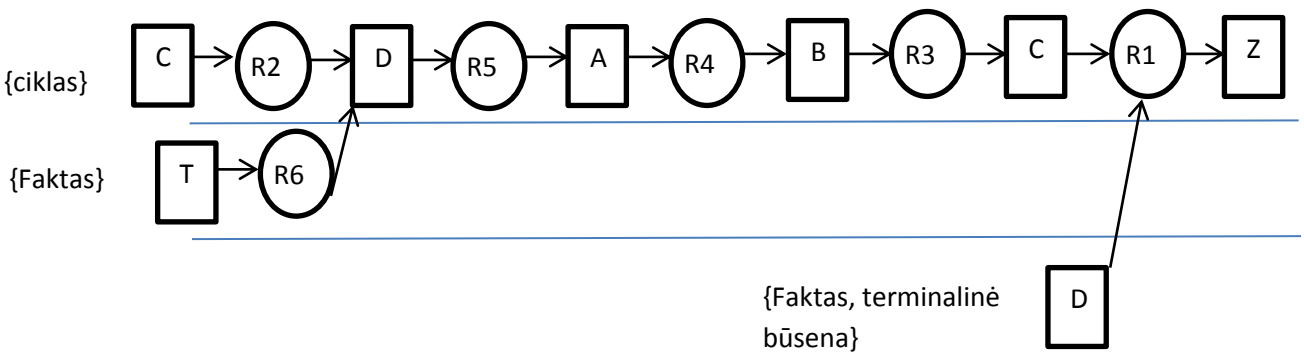
Semantinis grafas 1: Devynios produkcijos

Pavyzdys: modifikuotos devynios implikacijos

Dar kartelį bandomas algoritmo nuoseklumo teisingumas – tikrinama ar praktiniai algoritmo darbo rezulatai suderinami su teoriniu veikimo modeliu.

Duomenų failo turinys	Programos darbo protokolas

#Dainius Jocas
CDZ
CD
BC
AB
DA
TD
GA
HB
JC
 	
T
 	
Z
	List of implications that was read from the data file:
	  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
	Initial list of facts: {T}
	The goal for the program: Z
	
	Searching for: Z
	Derivation path: <- Z
	Searching for: C
	Derivation path: <- C <- Z
	Searching for: B
	Derivation path: <- B <- C <- Z
	Searching for: A
	Derivation path: <- A <- B <- C <- Z
	Searching for: D
	Derivation path: <- D <- A <- B <- C <- Z
	Searching for: C
	LOOP!!!
	Derivation path: <- D <- A <- B <- C <- Z
	Searching for: T
	T is in facts.
	D goes to list of facts!
	A goes to list of facts!
	B goes to list of facts!
	C goes to list of facts!
	Searching for: D
	D is in facts.
	Z goes to list of facts!
	
	The goal is reached!
	The answer is: {R6, R5, R4, R3, R1}



Semantinis grafas 2: Modifikuotos devynio produkcijos

Pavyzdys: neišvedamas tikslas

Duomenų failo turinys	Programos darbo protokolas

`# Dainius Jocas`
`AB`
 	
`A`
 	
`Z`
	`List of implications that was read from the data file:`
	`  R1: A -> B`
	`Initial list of facts: {A}`
	`The goal for the program: Z`
	
	`Searching for: Z`
	`DEADEND!`
	
	`Goal is not derivable in the production system`

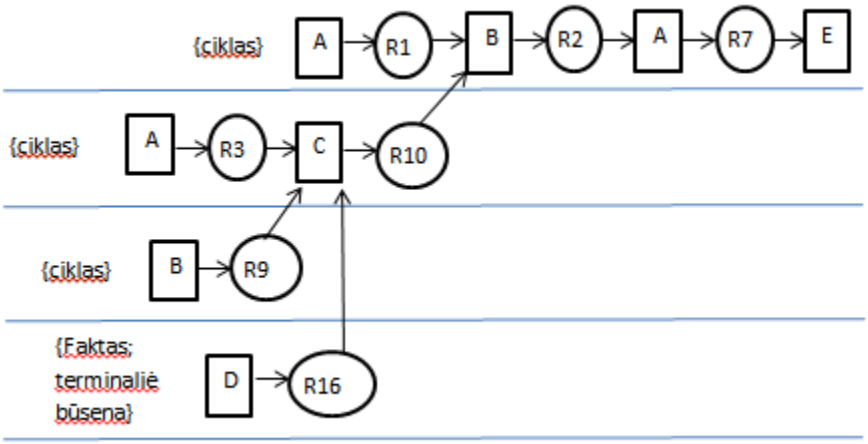
Pavyzdys: pilnasis grafas

Šiuo pavyzdžiu norima parodyti, kad pilnajame grafe svarbiausia yra gerai aptikli ciklines paieškas. Net Pilnojo grafo, kurio dydis yra 5, paieška užsiciklino net į tris ciklus.

Duomenų failo turinys	Programos darbo protokolas

`# Dainius Jocas`
`AB`
`BA`
`AC`
`CA`
`AD`
`DA`
`AE`
`EA`
`BC`
`CB`
`BD`
`DB`
`BE`
`EB`
`CD`
`DC`
`CE`
`EC`
`DE`
`ED`
	`List of implications that was read from the data file:`
	`  R1: A -> B`
	`  R2: B -> A`
	`  R3: A -> C`
	`  R4: C -> A`
	`  R5: A -> D`
	`  R6: D -> A`
	`  R7: A -> E`
	`  R8: E -> A`
	`  R9: B -> C`
	`  R10: C -> B`
	`  R11: B -> D`
	`  R12: D -> B`
	`  R13: B -> E`
	`  R14: E -> B`
	`  R15: C -> D`
	`  R16: D -> C`
	`  R17: C -> E`
	`  R18: E -> C`
	`  R19: D -> E`

D E	R20: E -> D Initial list of facts: {D} The goal for the program: E Searching for: E Derivation path: <- E Searching for: A Derivation path: <- A <- E Searching for: B Derivation path: <- B <- A <- E Searching for: A LOOP!!! Derivation path: <- B <- A <- E Searching for: C Derivation path: <- C <- B <- A <- E Searching for: A LOOP!!! Derivation path: <- C <- B <- A <- E Searching for: B LOOP!!! Derivation path: <- C <- B <- A <- E Searching for: D D is in facts. C goes to list of facts! B goes to list of facts! A goes to list of facts! E goes to list of facts! The goal is reached! The answer is: {R16, R10, R2, R7}



Semantinis grafas 3: Pilnasis grafas

Išvados

Šiame kurso „Dirbtinis intelektas“ laboratoriniame darbe buvo suprogramuota atbulinio išvedimo produkcijų sistemoje demostracinė sistema Java programvimo kalba. Programos veikimas ištestuotas. Rezultatų teisingumas patikrintas juos lyginant su semantiniais grafais.

Tolimesni sistemos plėtojimas galėtų būti algoritmo pritaikymas sprendžiant matematinės logikos uždavinius.

Literatūra

[Čyr08] V. Čyras. Intelektualios sistemos, paskaitų konspektas. URL: http://uosis.mif.vu.lt/~cyras/AI/konspektas-intelektualios-sistemos.pdf.

[Nea86] R. E. Neapolitan. Forward-chaining versus a graph approach as the inference engine in expert systems. URL: http://uosis.mif.vu.lt/~cyras/AI/Neapolitan-1986-in-Principles_of_Expert_Systems.pdf

[Neg05] M. Negnevitsky. Artificial Intelligence: A Guide to Intelligent Systems.