**ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ**

**ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԿԻՐԱՌԱԿԱՆ ՄԱԹԵՄԱՏԻԿԱՅԻ ՖԱԿՈՒԼՏԵՏ**

**Ծրագրավորման և Ինֆորմացիոն տեխնոլոգիաների ամբիոն**

**ՀԱՇՎՈՂԱԿԱՆ ՄԵՔԵՆԱՆԵՐԻ ՀԱՄԱԼԻՐՆԵՐԻ ՀԱՄԱԿԱՐԳԵՐԻ ԵՎ ՑԱՆՑԵՐԻ ՄԱԹԵՄԱՏԻԿԱԿԱՆ ԵՎ ԾՐԱԳՐԱՅԻՆ ԱՊԱՀՈՎՈՒՄ**

**Ոսկանյան Վահագն Գևորգի**

**ՄԱԳԻՍՏՐՈՍԱԿԱՆ ԹԵԶ**

**ԱՌԱՋԻՆ ԿԱՐԳԻ ՏՐԱՄԱԲԱՆՈՒԹՅԱՆ ՄԵՋ ՌԵԶՈԼՈՒՑԻԱՅԻ ՀԱՄԱՐ ԼԻՏԵՐԱԼՆԵՐԻ ԸՆՏՐՈՒԹՅԱՆ ԲԱՐԵԼԱՎՈՒՄ ՄԵՔԵՆԱՅԱԿԱՆ ՈՒՍՈՒՑՄԱՆ ՄԻՋՈՑՈՎ**

***«Տեղեկատվական տեխնոլոգիաներ» մասնագիտությամբ***

***Ինֆորմատիկայի և կիրառական մաթեմատիկայի մագիստրոսի որակավորման աստիճանի հայցման համար***

**ԵՐԵՎԱՆ 2025**

***Ուսանող`****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*ստորագրություն*

***Ոսկանյան Վահագն***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ազգանուն, անուն

***Գիտական ղեկավար՝\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

*ստորագրություն*

***ֆ․մ.գ.թ. , դոցենտ, Հովհաննես Բոլիբեկյան***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

գիտ. աստիճան, կոչում, ազգանուն, անուն

***«Թույլատրել պաշտպանության»***

***Ամբիոնի վարիչ`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

ստորագրություն

***ֆ․մ.գ.թ. , Սարգսյան Ս․***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

գիտ. աստիճան, կոչում, ազգանուն, անուն

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025թ.

# ՀԱՄԱՌՈՏԱԳԻՐ

ԱՌԱՋԻՆ ԿԱՐԳԻ ՏՐԱՄԱԲԱՆՈՒԹՅԱՆ ՄԵՋ ՌԵԶՈԼՈՒՑԻԱՅԻ ՀԱՄԱՐ ԼԻՏԵՐԱԼՆԵՐԻ ԸՆՏՐՈՒԹՅԱՆ ԲԱՐԵԼԱՎՈՒՄ ՄԵՔԵՆԱՅԱԿԱՆ ՈՒՍՈՒՑՄԱՆ ՄԻՋՈՑՈՎ

УЛУЧШЕНИЕ ВЫБОРА ЛИТЕРАЛОВ В РЕЗОЛЮЦИИ ЛОГИКИ ПЕРВОГО ПОРЯДКА С ИСПОЛЬЗОВАНИЕМ МАШИННОГО ОБУЧЕНИЯ

IMPROVING LITERAL SELECTION IN FIRST-ORDER LOGIC RESOLUTION USING MACHINE LEARNING

Այս աշխատանքն ուսումնասիրում է առաջին կարգի տրամաբանության մեջ ռեզոլյուցիայի մեթոդի արդյունավետության բարձրացմանը՝ կենտրոնանալով լիտերալների օպտիմալ ընտրության վրա մեքենայական ուսուցման միջոցով:  Ռեզոլյուցիան, որպես ավտոմատ ապացուցման հիմնական գործիք, հաճախ բախվում է հաշվողական բարդության խնդիրների՝ պայմանավորված լիտերալների ոչ ճիշտ ընտրությամբ: Ուսումնասիրությունը նպատակ ունի մշակել նոր մոտեցում, որը կօգտագործի մեքենայական ուսուցման ալգորիթմներ՝ ռեզոլյուցիայի ընթացքում լիտերալների ավելի արդյունավետ ընտրություն ապահովելու համար: Այն կնվազեցնի որոնման տարածությունը, կբարելավի ապացուցման արագություն և ապացույց գտնելու հնարավորությունը։

Աշխատանքի արդյունքները կարող են կիրառվել թեորեմներ ավտոմատ ապացուցող համակարգերում՝ բարելավելով դրանց արտադրողականությունը:

Contents

[ՀԱՄԱՌՈՏԱԳԻՐ 3](#_Toc196513284)

[ՆԵՐԱԾՈՒԹՅՈՒՆ 5](#_Toc196513285)

[Գլուխ 1 6](#_Toc196513286)

[1.1 Դևիսի և Փաթնեմի մեթոդը 6](#_Toc196513287)

[1.2 Ռեզոլյուցիայի մեթոդը տրամաբանակ արտահայտություններում 7](#_Toc196513288)

[1.3 Փոխարինում և ունիֆիկացիա 9](#_Toc196513289)

[1.4 Ունիֆիկացման ալգորիթմ 11](#_Toc196513290)

[1.5 Ռեզոլյուցիայի մեթոդը առաջին կարգի տրամաբանական արտահայտությունների համար 14](#_Toc196513291)

[**Գլուխ 2** 15](#_Toc196513292)

# ՆԵՐԱԾՈՒԹՅՈՒՆ

Ներկայացվում է առաջին կարգի տրամաբանության մեջ ռեզոլյուցիայի մեթոդի լիտերալների ընտրության օպտիմալացման խնդիրը՝ մեքենայական ուսուցման մեթոդների կիրառմամբ։ Ուսումնասիրության արդիականությունը պայմանավորված է ավտոմատ ապացուցման համակարգերի արդյունավետության բարձրացման անհրաժեշտությամբ, հատկապես բարդ տրամաբանական խնդիրների լուծման համատեքստում։

Աշխատանքի նպատակն է մշակել լիտերալների ընտրության նոր մոտեցում, որն օգտագործում է մեքենայական ուսուցման ալգորիթմներ՝ ռեզոլյուցիայի արդյունավետությունը բարելավելու համար։ Հիմնական խնդիրները ներառում են՝ ռեզոլյուցիայի ընթացքում լիտերալների ընտրության օպտիմալ ռազմավարության մշակումը, մեքենայական ուսուցման մոդելի ստեղծումը, որը կկանխատեսի լիտերալների ամենահարմար զույգերը, և մեթոդի փորձարկումը ստանդարտ տրամաբանական խնդիրների վրա։

Ուսումնասիրության օբյեկտը ռեզոլյուցիայի մեթոդն է առաջին կարգի տրամաբանության մեջ, իսկ առարկան՝ լիտերալների ընտրության օպտիմալացումը մեքենայական ուսուցման միջոցով։ Աշխատանքի վարկածն այն է, որ մեքենայական ուսուցման մոդելի կիրառումը կբարելավի ռեզոլյուցիայի արդյունավետությունը՝ նվազեցնելով որոնման տարածությունը և ապացուցման ժամանակը։

# Գլուխ 1

## 1.1 Դևիսի և Փաթնեմի մեթոդը

Ենթադրենք՝ -ը դիզյունկտների բազմություն է։ Մեթոդը, ըստ էության, բաղկացած է հետևյալ չորս կանոններից`

1. *Տավտոլոգիայի կանոն՝* -ից ջնջում ենք բոլոր տավտոլոգիա հիմնական դիզյունկտները։ Մնացած բազմությունը անհամատեղելի է, այն և միայն այն դեպքում, եթե -ը անհամատեղելի է:
2. *Մեկ լիտերալ դիզյունկտների կանոն՝* եթե -ում գոյություն ունի մեկ լիտերալ պարունակող հիմնական դիզյունկտ , ապա -ը ստացվում է -ից՝ ջնջելով այն հիմնական դիզյունկտները, որոնք պարունակում են : Եթե -ը դատարկ է, ապա -ը համատեղելի է: Հակառակ դեպքում, կառուցում ենք -ը՝ -ից ջնջելով -ի մուտքերը: -ը անհամատեղելի է, այն և միայն այն դեպքում, եթե -ը նույնպես անհամատեղելի է: Նշենք, որ եթե -ը մեկ լիտերալ հիմնական դիզյունկտ է, ապա այն ջնջելիս կվերածվի -ի։
3. *Մաքուր լիտերալների կանոն`* -ի հիմնական դիզյունկտում գտնվող լիտերալը կոչվում է *մաքուր*-ում, այն և միայն այն դեպում, եթե -ը չի հանդիպում -ի որևէ հիմնական դիզյունկտում: Եթե -ը մաքուր լիտերալ է, ապա ջնջում ենք բոլոր հիմնական դիզյունկտները, որոնք պարունակում են ։ Մնացած բազմությունը անհամատեղելի է, այն և միայն այն դեպքում, եթե-ը անհամատեղելի է:
4. *Բաժանման կանոն`* եթե  բազմությունը կարելի է ներկայացնել հետևյալ տեսքով՝ , որտեղ -ի​ն և -ը ազատ են -ից և -ից, ապա ստանում ենք երկու բազմություն՝  *և* , -ը անհամատեղելի է, այն և միայն այն դեպքում, երբ *-*ը անհամատեղելի է, այսինքն՝ և -ը, և ​-ը անհամատեղելի են:

Վերոհիշյալ կանոնները շատ կարևոր են: Հետագայում կտեսնենք, որ այս կանոններն ունեն ավելի լայն կիրառություն: Բերենք օրինակներ՝ այս կանոնների օգտագործումը ցույց տալու համար:

Օրինակ՝ ցույց տանք, որ -ը անհամատեղելի է:

Քանի, որ վերջնական բանաձևը պարունակում է դատարկ դիզյունկտ , ապա -ը անհամատեղելի է:

Օրինակ՝ ցույց տանք, որ -ը համատեղելի է:

Քանի որ բաժանման երկու բազմություններն էլ համատեղելի են, ապա -ը նույնպես համատեղելի է:

Օրինակ՝ ցույց տանք, որ -ը համատեղելի է:

Այսպիսով -ը համատեղելի է։

## 1.2 Ռեզոլյուցիայի մեթոդը տրամաբանակ արտահայտություններում

Ռեզոլյուցիայի մեթոդը, ըստ էության, [Դևիսի և Փաթնեմի](#_1.1_Դևիսի_և) մեկ լիտերալ դիզյունկտների կանոնի ընդհանրացումն է:

Օրինակ դիտարկենք հետևալ դիզունկտները՝

Օգտագործելով մեկ լիտերալ դիզունկտների կանոնը, -ից և -ից մենք կարող ենք ստանալ նոր դիզյունկտ

Մեկ լիտերալ դիզյունկտների կանոնը մեզ անհրաժեշտ է, որպեսզի նախ որոշենք, արդյոք կա լիտերալների հակադիր զույգ (օրինակ՝ ) -ում և (օրինակ՝  ) -ում, ապա ջնջենք այդ զույգը -ից և -ից, որպեսզի ստանանք նոր դիզյունկտ , որը -ն է:

Վերոհիշյալ կանոնը ընդհանրացնելով և այն կիրառելով դիզյունկտների ցանկացած զույգի նկատմամբ (ոչ պարտադիր միայն մեկ լիտերալ պարունակող), մենք ստանում ենք հետևյալ կանոնը, որը կանվանենք ***ռեզոլյուցիայի կանոն***:

Ցանկացած երկու դիզյունկտների համար՝ և , եթե գոյություն ունի ​լիտերալ -ում, որը հակադիր է ​ լիտերալին -ում, ապա ջնջելով ​-ը -ից և ​-ը -ից, մենք կառուցում ենք մնացած դիզյունկտների դիզյունկցիան: Ստացված դիզյունկտը կոչվում է -ի և -ի *ռեզոլվենտ*:

Օրինակ դիտարկենք հետևյալ դիզյունկտները՝

-ը պարունակում է լիտերալ, որը հակադիր է -ում գտնվող լիտերալին: Ուստի, ջնջելով -ն -ից և -ն -ից, մենք կառուցում ենք մնացած դիզյունկտների դիզյունկցիան`, ստացված ռեզոլվենտը կլինի ։

Ռեզոլվենտի կարևոր հատկությունն այն է, որ ցանկացած ռեզոլվենտ, որը ստացվում է երկու դիզյունկտներից՝  և , -ի և -ի տրամաբանական հետևանքն է: Այս հատկությունը հաստատվում է հետևյալ թեորեմով`

**Թեորեմ 1.0:** Եթե տրված են երկու դիզյունկտներ՝ և , ապա -ի և -ի ռեզոլվենտը -ն -ի և -ի տրամաբանական հետևանքն է:

Ապացույց՝ ենթադրենք , և , որտեղ և -ը լիտերալների դիզյունկցիաներ են: Ենթադրենք, որ -ը և -ը ճշմարիտ են ինտերպրետացիայում: Մենք ցանկանում ենք ապացուցել, որ -ի և -ի ռեզոլվենտը՝ -ն, նույնպես ճշմարիտ է -ում: Ապացույցի համար նշենք, որ -ը կամ -ը կեղծ են -ում։ Եթե -ը կեղծ է -ում, ապա -ը կարող է ճշմարիտ լինել միայն այն դեպքում, եթե ​-ը ճշմարիտ է -ում: Նույն կերպ, եթե -ը կեղծ է -ում, ապա -ը կարող է ճշմարիտ լինել միայն այն դեպքում, եթե -ը ճշմարիտ է -ում։ Ոեզոլվենտը՝ ​, կլինի ճշմարիտ -ում, եթե ​-ը կամ -ը ճշմարիտ է -ում: Քանի որ ​-ը կամ -ը պետք է ճշմարիտ լինեն -ում, ապա -ն նույնպես ճշմարիտ է-ում։ Դա այն է, ինչ պետք էր ապացուցել:

**Սահմանում`** Ենթադրենք՝ -ը դիզյունկտների բազմություն է: -ից -ի ռեզոլյուցիոն արտածումը դիզյունկտների վերջավոր հաջորդականություն է՝ որտեղ յուրաքանչյուր ​-ն կամ պատկանում է -ին, կամ նախորդ դիզյունկտների ռեզոլվենտն է, և : -ից (դատարկ դիզյունկտ) արտածումը կոչվում է -ի հերքում (կամ -ի անհամատեղելիության ապացույց):

Մենք ասում ենք, որ դիզյունկտը կարող է արտածվել կամ ստացվել -ից, եթե գոյություն ունի -ի արտածում -ից:

Օրինակ դիտարկենք բազմություն՝

(1)-ից և (2)-ից կարող ենք ստանալ ռեզոլվենտ՝ : (4)-ից և (3)-ից կարող ենք ստանալ ռեզոլվենտ՝ ։ Քանի, որ -ն ստացվում է -ից ռեզոլյուցիայի կանոնի կիրառմամբ, ապա համաձայն թեորեմ 1.0-ի, -ը -ի տրամաբանական հետևանքն է: Ուստի, -ը անհամատեղելի է։

## 1.3 Փոխարինում և ունիֆիկացիա

Մենք դիտարկեցինք *ռեզոլյուցիայի մեթոդը* տրամաբանական արտահայտությունների համար: Հաջորդիվ մենք այն կտարածենք *առաջին կարգի տրամաբանության* վրա: Նշել ենք, որ ռեզոլյուցիայի կանոնի կիրառման հիմնական պահը հակադիր լիտերալների գտնելն է երկու դիզյունկտներում: Երբ դիզյունկտները չեն պարունակում փոփոխականներ, ապա դա շատ պարզ է: Սակայն, երբ դիզյունկտները պարունակում են փոփոխականներ, ապա խնդիրը բարդանում է: Օրինակի համար դիտարկենք հետևյալ դիզյունկտները՝

Չկա որևէ լիտերալ ​-ում, որը հակադիր լինի -ի որևէ լիտերալի: Սակայն, եթե մենք ​-ում x-ը փոխարինենք -ով, իսկ ​-ում x-ը փոխարինենք -ն, ապա կստանանք՝

Գիտենք, որ ​-ը և ​-ը համապատասխանաբար ​-ի և ​-ի հիմնական օրինակներն են, իսկ -ն և -ն հակադիր են միմյանց: Ուստի, -ից և ​-ից մենք կարող ենք ստանալ ռեզոլվենտ՝

Ընդհանուր դեպքում, եթե ​-ում x-ը փոխարինենք -ով, ապա կստանանք՝

Կրկին ​-ը ​-ի օրինակ է: Միևնույն ժամանակ, ​-ում -ը հակադիր է ​-ում -ին: Ուստի, մենք կարող ենք ստանալ ռեզոլվենտ ​-ից և ​-ից:

-ը ​-ի օրինակ է: Փոփոխականները ​-ում և -ում համապատասխան թերմերով փոխարինելով, ինչպես նշված է վերևում, մենք կարող ենք ստեղծել նոր դիզյունկտներ ​-ից և ​-ից: Բացի այդ, ​-ը *ամենաընդհանուր դիզյունկտն* է այն իմաստով, որ վերը նշված գործընթացով ստացված բոլոր այլ դիզյունկտները -ի օրինակներ են: ​-ը նույնպես կանվանենք -ի և ​-ի ռեզոլվենտ:

**Սահմանում`** *փոխարինումը* (substitution) վերջավոր բազմություն է՝ , որտեղ՝ յուրաքանչյուր -ն փոփոխական է, յուրաքանչյուր -ն թերմ է, որը տարբերվում է -ից, բոլոր ​-երը տարբեր են: Եթե -ը հիմնական թերմեր են (այսինքն՝ չեն պարունակում փոփոխականներ), ապա փոխարինումը կոչվում է հիմնական փոխարինում: Փոխարինումը, որը չի պարունակում որևէ տարր, կոչվում է դատարկ փոխարինում և նշանակվում է -ով: Փոխարինումը գրելու համար մենք կօգտագործենք հունարեն տառեր (օրինակ՝ ):

Օրինակ հետևալ երկու բազմությունները հանդիսանում են փոխարինում՝

**Սահմանում`** ենթադրենք -ը փոխարինում է, և -ն արտահայտություն է: Այդ դեպքում -ն արտահայտություն է, որը ստացվում է -ից՝ -ում *​-*ի բոլոր հանդիպումները միաժամանակ փոխարինելով -ով: -ն կոչվում է -ի *օրինակ*: (Նշենք, որ օրինակի այս սահմանումը համատեղելի է գլուխ 4-ում տրված դիզյունկտի հիմնական օրինակի սահմանման հետ:)

Օրինակ՝ ենթադրենք և: Այդ դեպքում։

**Սահմանում`** ենթադրենք և երկու փոխարինումներ են: Այդ դեպքում -ի և -ի **կոմպոզիցիան** (նշանակում ենք ) այն փոխարինումն է, որը ստացվում է հետևալ բազմությունից՝

Ջնջելով բոլոր այն -երը որոնց համար , և բոլոր -երը որոնց համար (այսինքն՝  ​-ն արդեն առկա է -ում)։

Օրինակ ՝ ենթադրենք

Այդ դեպքում՝ ։ Սակայն, քանի որ (այսինքն ), ապա պետք է հեռացնել բազմությունից։ Բացի այդ, քանի որ -ը և ​-ը առկա են -ում, ապա ​-ը և ​-ը (այսինքն՝  -ը և -ը) նույնպես պետք է հեռացվեն: Այսպիսով, մենք ստանում ենք՝

**Սահմանում`** փոխարինումը -ն կոչվում է ունիֆիկատոր (unifier) բազմության համար, այն և միայն այն դեպքում, երբ ։ Ասում են, որ բազմությունը *ունիֆիկացվող* է, եթե բազմության համար գոյություն ունի ունիֆիկատոր:

**Սահմանում`** ունիֆիկատոր -ն բազմության համար կոչվում է ամենաընդհանուր ունիֆիկատոր (most general unifier, MGU), այն և միայն այն դեպքում, երբ ցանկացած այլ ունիֆիկատոր -ի համար գոյություն ունի փոխարինում , այնպես որ՝ ։

Օրինակ՝ բազմությունը ունիֆիկացվող է քանի, որ հանդիսանում է ունիֆիկատոր նրա համար։

## 1.4 Ունիֆիկացման ալգորիթմ

Այս պարբերությունում կներկայացնենք ունիֆիկացման ալգորիթմ, որը թույլ է տալիս գտնել ամենաընդհանուր ունիֆիկատորը վերջավոր ունիֆիկացվող բազմության համար: Եթե բազմությունը չի ունիֆիկացվում, ալգորիթմը կհայտնաբերի նաև այդ փաստը:

**Սահմանում`** ոչ դատարկ արտահայտությունների բազմության -ի անհամապատասխանությունների բազմությունը ստացվում է գտնելով առաջին (ձախից) դիրքը, որտեղ -ի բոլոր արտահայտությունները չունեն նույն սիմվոլը, այնուհետև յուրաքանչյուր արտահայտությունից դուրս գրելով այն ենթաարտահայտությունը, որը սկսվում է այդ դիրքում գտնվող սիմվոլից։ Այս ենթաարտահայտությունների բազմությունը կոչվում է -ի *անհամապատասխանությունների բազմություն*:

Օրինակ՝ եթե -ն հետևյալ բազմությունն է՝ , ապա առաջին դիրքը, որտեղ -ի բոլոր արտահայտությունները չունեն նույն սիմվոլը, հինգերորդ դիրքն է, քանի որ բոլոր արտահայտությունները ունեն նույն առաջին չորս սիմվոլները՝ : Այսպիսով, անհամապատասխանությունների բազմությունը բաղկացած է համապատասխան ենթաարտահայտություններից, որոնք սկսվում են հինգերորդ դիրքից, և դա հետևյալ բազմությունն է՝ ։

*Ունիֆիկացման ալգորիթմ`*

*Քայլ* 1. Բազմություններ ։

*Քայլ* 2. Եթե -ն միալիտերալ դիզյունկտ է, ապա -ն -ի *ամենաընդհանուր ունիֆիկատորն* է: Հակառակ դեպքում, գտնել -ի անհամապատասխանությունների բազմությունը​:

*Քայլ* 3. Եթե -ում գոյություն ունեն և տարրեր, այնպիսին որ -ն փոփոխական է, որն չի հայտնվում -ում, ապա անցնել քայլ 4-ին: Հակառակ դեպքում, ավարտել՝ -ն չի ունիֆիկացվում:

*Քայլ* 4. Սահմանել և : (Նշենք, որ )։

*Քայլ* 5. -ին վերագրել  արժեքը և անցնել քայլ 2-ին:

Օրինակ՝ գտնել ամենաընդհանուր ունիֆիկատորը

1. և ։ Քանի, որ ​-ն միալիտերալ դիզյունկտ չէ, ուստի ​-ն -ի ամենաընդհանուր ունիֆիկատոր չէ:
2. Անհամապատասխանությունների բազմությունը՝ : -ում գոյություն ունի փոփոխական : որը չի հանդիպում *-*ում։
3. Սահմանենք՝

,

1. ​-ը միալիտերալ դիզյունկտ չէ, քանի որ գտնվել է անհամապատասխանությունների բազմություն -ի համար։
2. -ից կգտնենք և *։*
3. Սահմանենք՝

,

1. -ը միալիտերալ դիզյունկտ չէ, քանի որ գտնվել է անհամապատասխանությունների բազմություն -ի համար։ : -ից կգտնենք և :
2. Սահմանենք՝

,

1. Քանի, որ​-ը միալիտերալ դիզյունկտ է, ապա -ն -ի ամենաընդհանուր ունիֆիկատորն է:

**Թեորեմ 1.1:** (Ունիֆիկացման թեորեմ)՝ եթե-ն վերջավոր ոչ դատարկ ունիֆիկացվող արտահայտությունների բազմություն է, ապա ունիֆիկացման ալգորիթմը միշտ կավարտվի քայլ 2-ում, և վերջին -ն կլինի -ի ամենաընդհանուր ունիֆիկատորը:

## 1.5 Ռեզոլյուցիայի մեթոդը առաջին կարգի տրամաբանական արտահայտությունների համար

Նախորդ պարբերությունում ներկայացված ունիֆիկացման ալգորիթմի շնորհիվ մենք կարող ենք այժմ դիտարկել առաջին կարգի տրամաբանության համար ռեզոլյուցիայի մեթոդը:

**Սահմանում`** եթե դիզյունկտ -ի երկու կամ ավելի լիտերալներ (նույն նշանով) ունեն ամենաընդհանուր ունիֆիկատոր , ապա -ն կոչվում է -ի *սոսնձում*: Եթե -ն միալիտերալ դիզյունկտ է, ապա սոսնձումը կոչվում է *միալիտերալ սոսնձում*:

Օրինակ՝ ենթադրենք : Այդ դեպքում առաջին և երկրորդ լիտերալները (ընդգծված) ունեն ամենաընդհանուր ունիֆիկատոր : Ուստի, -ը -ի սոսնձումն է:

**Սահմանում`** եթե ​-ը և ​-ը երկու դիզյունկտներ են (կոչվում են դիզյունկտ-նախադրյալներ), որոնք չունեն ընդհանուր փոփոխականներ: Թող ​-ը և ​-ը լինեն երկու լիտերալներ համապատասխանաբար ​-ում և ​-ում: Եթե -ը և ​-ն ունեն ամենաընդհանուր ունիֆիկատոր , ապա դիզյունկտը՝

կոչվում է -ի և -ի (երկուական) ռեզոլվենտ: ​-ը և ​-ը կոչվում են *կրճատվող լիտերալներ*:

Օրինակ՝ ենթադրենք , : Քանի որ -ը ներառված է և՛ ​-ում, և՛ ​-ում, մենք փոխարինում ենք ​-ում -ը -ով՝ ։ Ընտրում ենք՝ , : Քանի որ :, ապա ​-ը և ​-ն ունեն ամենաընդհանուր ունիֆիկատոր: Այսպիսով՝

Ուստի, *-*ն -ի և ​-ի երկուական ռեզոլվենտն է։ -ը և -ն կրճատվող լիտերալներն են:

**Գլուխ 2**

## 2.1 TPTP գրադարանի նկարագրություն

**TPTP** ( *Thousands of Problems for Theorem Provers* — «Հազարավոր խնդիրներ թեորեմներ ապացուցող ծրագրերի համար») միավորում է ավելի քան 10.000 փորձնական խնդիր՝ նախատեսված ավտոմատ ապացուցման (ATP) համակարգերի կատարողականությունը չափելու, համեմատելու և զարգացնելու նպատակով։ Գրադարանը ստեղծվել է 1993-ին, պահպանվում է Մայամիի համալսարանում (Geoff Sutcliffe) և նոր թողարկումներով ընդլայնվում է մինչ օրս։

TPTP-ն ընդգրկում է տրամաբանական խնդիրներ տարբեր ոլորտներից (մաթեմատիկական վկայություններ, ծրագրա­վորման լեզուների վերլուծություն, արհեստական բանականություն, խմբերով տեսություն և այլն) և տարբեր բարդության մակարդակների՝ ապահովելով համաչափ «դյուրին → չափավոր → դժվար» սանդղակ։ Յուրաքանչյուր ֆայլ պարունակում է՝

* **մետատվյալներ** - եզակի կարճ անուն, թեմատիկ դաս, դժվարության գնահատական, աղբյուր,
* **խնդրի ձևակերպում** ընտրված TPTP ֆորմատներից մեկով,
* **ակսիոմների** և **հետազոտվող վարկածի** բաժանում։

TPTP լեզուն Prolog-ի սինթաքսի ընդլայնում է։ նախադասությունները գրվում են *annotated formula* տեսքով՝ fof(<անուն>, <դեր>, <ֆորմուլա>), cnf(<անուն>, <դեր>, <կլաուզա>), tff / thf և այլն։

* **CNF** (clause normal form) - կլաուզային նորմալ ձև,
* **FOF** (first-order form) - չտիպավորված առաջին կարգ,
* **TFF** (typed first-order form) - տիպավորված առաջին կարգ,
* **TXF** (extended first-order form) - ընդլայնված առաջին կարգ՝ ներկառուցված թվաբանական տիպերով և ֆունկցիաներով։

Յուրաքանչյուր տողի *role* դաշտը (axiom, conjecture, lemma, definition…) հնարավորություն է տալիս թեստային հավաքածուն ճշգրիտ բաժանել ակսիոմների և ապացուցելի թեզի, ինչը պարտադիր է ռեզոլյուցիոն ATP-ների համար։

Մենք ծրագրի տվյալների բազա ստեղծելու համար օգտվելու ենք այս գրադարանի ակսիոմների հավաքածուից և հիմնականում CNF,FOF ֆորմատներից։

## 2.2 Vampire ATP համակարգի նկարագրություն

**Vampire-ը** (Vampire Automated Theorem Prover) ավտոմատ թեորեմներ ապացուցող (ATP) առաջատար համակարգերից մեկն է, որը մշակվել է Մանչեստերի համալսարանում։ Այն հատկապես հայտնի է իր բարձր արդյունավետությամբ առաջին կարգի տրամաբանության (first-order logic, FOL) և ավելի բարդ ֆորմալ համակարգերի համար։ Vampire-ն օգտագործում է ռեզոլյուցիայի (resolution), սուպերպոզիցիայի (superposition) և այլ ժամանակակից մեթոդներ՝ թեորեմների ապացուցման համար, ինչը հնարավորություն է տալիս աշխատել ինչպես մաթեմատիկական, այնպես էլ հաշվողական տրամաբանության բարդ խնդիրների հետ։

Այն աշխատում է TPTP-ի բոլոր հիմնական ֆորմատներով, այսինք խնդիրը հնարավոր է ծրագիր մուտքագրել այդ ֆորմատով։ Ծրագիրը հենվում է ժամանակակից ապացուցման ռազմավարությունների վրա, որոնք հատուկ կիրառելի են TPTP-ում ներկայացված խնդիրների համար։ Այն բազմիցս ճանաչվել է ամենաարդյունավետ ATP համակարգերից մեկը CASC (Conference on Automated Deduction) մրցույթներում։

Մենք կօգտագործենք այս ծրագիրը ստեղծված տվյալների բազայի խնդիրները լուծելու համար և այլ նպատակներով։

## 2.3 Տվյալների բազաի ստեղծում

սդսդ

պարունակում

պարունակում