# Процедура резолюций (Апальков Ф.С. иу7-33м 2022)

## Запуск

import Athom from "./Resolutions/Athom.js";

import Disjunction from "./Resolutions/Disjunction.js";

import { absorption } from "./Resolutions/createResolventa.js";

import createResolventa from './Resolutions/createResolventa.js'

let a = new Athom('A', true)

let b = new Athom('B', true)

let c = new Athom('C', true)

let na = new Athom('A', false)

let nb = new Athom('B', false)

let nc = new Athom('C', false)

let d = new Athom('D', true)

let nd = new Athom('D', false)

let ax1 = new Disjunction([a, b])

let ax2 = new Disjunction([na, c])

let ax3 = new Disjunction([nb, d])

let ax4 = new Disjunction([nc])

let ax5 = new Disjunction([nd])

let ax6 = new Disjunction([na])

import resolution from './Resolutions/resolution.js'

let r = resolution([ax1, ax2, ax3, ax4, ax5], ax6, 'full', 1000)

console.log('all')

## Атом

class Athom{

    constructor(name, sign){

        this.name = name

        this.sign = sign

    }

}

export default Athom

## Дизъюнкция

class Disjunction{

    constructor(listOfAthoms){

        this.Athoms = listOfAthoms

    }

}

export default Disjunction

## Приведение дизъюнкции к виду без повторяющихся атомов или получение из нее 0 или 1

/\*\*

 \*

 \* @param {Disjunction} dis - дизъюнкт из которого удаляем дубли

 \* @return {Disjunction} - дизъюнкт или false \ true

 \*/

function absorption(dis){

    let arr = []

    for(let at of dis.Athoms){

        // провереям есть ли такой же атом

        let flag = true

        for(let inat of arr){

            if(inat.name == at.name){

                if(inat.sign == at.sign){

                    // если знаки и мена совпали не добавляем этот атом в результат

                    flag = false

                }

                else{

                    // если имена совпали а знаки разные значит он просто 1

                    return true

                }

            }

        }

        if (flag){

            // если его надо добавить то добавляем атом

            arr.push(at)

        }

    }

    return new Disjunction(arr)

}

## Создание дизъюнкта резольвенты

/\*\*

 \*

 \* @param firstDis - первый дизъюнкт

 \* @param secondDis - второй дизъюнкт

 \* @returns резольвент. дизъюнкт, null - если не найдено контраарной пары или false если был пустой дизъюнкт

 \*/

function createResolventa(firstDis, secondDis){

    firstDis = absorption(firstDis)

    secondDis = absorption(secondDis)

    let firstAt = [ ... firstDis.Athoms ]

    let secondAt = [... secondDis.Athoms]

    if(firstDis && secondDis){

        // если обе не 0

        let bufAthom // Атом из первого списка для которого ищем противоположный по знаку

        let flag = false

        // бежим по всем парам атомов и если имена совпали а знаки разные flag <- true

        cikle: for(bufAthom of firstAt){

            for(let inat of secondAt){

                if(bufAthom.name == inat.name){

                    if(bufAthom.sign != inat.sign){

                        flag = true

                        break cikle

                    }

                }

            }

        }

        if(flag){

            // если найдена противополажная пара атомов

            // заполнить список остальных атомов не включая в него найденные противоположности

            let leftAt = firstAt.filter(item => item.name != bufAthom.name)

            let rightAt = secondAt.filter(item => item.name != bufAthom.name)

            let retAthoms = [...leftAt , ...rightAt]

            let r = absorption( new Disjunction(retAthoms) )

            if(r.Athoms.length) return r

            else return false

        }

        else{

            // противоположных пар не найдено

            // вернуть null

            return null

        }

    }

    else{

        //  если кто-то 0

        return false

    }

}

export default createResolventa

## процедура резолюции

import fullResolution from "./resolutionVariants/full.js"

import basic from "./resolutionVariants/basic.js"

/\*\*

 \*

 \* @param axioms массив аксиом

 \* @param targetInverted инвертированная цель

 \* @param resolutionVariant тип обхода - full, basic (полный перебор, опорное мн-во)

 \* @param maxIterations

 \* @returns структура со статусом и результатом

 \*/

function resolution(axioms, targetInverted, resolutionVariant, maxIterations){

    if(resolutionVariant == 'full'){

        return fullResolution(axioms, targetInverted, maxIterations)

    }

    if(resolutionVariant == 'basic'){

        return basic(axioms , targetInverted , maxIterations)

    }

}

export default resolution

## Вариант полного перебора

/\*\*

 \*

 \* @param axioms массив аксиом

 \* @param targetInverted инвертированная цель

 \* @param maxIterations

 \* @returns структура со статусом и результатом

 \*/

function fullResolution(axioms, targetInverted, maxIterations){

    let DisjunctionStack = [...axioms, targetInverted]

    let iter = 0

    for(let i1 = 0; i1 < DisjunctionStack.length - 1; i1++){

        for(let i2 = 0; i2 < DisjunctionStack.length; i2++){

            // проходим по всем парам дизъюнктов

            if(i1 == i2) continue

            iter++

            if(iter > maxIterations) return {result: false, resStatus: 'выход по числу итенраций'}

            let dis1 = DisjunctionStack[i1]

            let dis2 = DisjunctionStack[i2]

            // пытаемся найти резольвенту

            let resolventa = createResolventa(dis1, dis2)

            if(resolventa instanceof Disjunction){

                DisjunctionStack.push(resolventa)

                // если получилось, то добавляем ее в стэк

            }

            if(resolventa == false){

                return {result: true, resStatus: 'доказано, найден ложный дизъюнкт'}

                // если нашли пустой дизъюнкт - то доказали утверждение

            }

            // если резолв. дизъ. был 1 , то мы его не добавляем

        }

    }

    return {result: false, resStatus: 'не найдено доказательство'}

}

## Вариант опорного множества

/\*\*

 \*

 \* @param axioms массив аксиом

 \* @param targetInverted инвертированная цель

 \* @param maxIterations

 \* @returns структура со статусом и результатом

 \*/

function basic(axioms, targetInverted, maxIterations){

    let S1 = [...axioms]

    let S2 = [targetInverted]

    let iter = 0

    while(true){

        let newDis = []

        for(let dis2 of S2){

            for(let dis1 of S1){

                // та же логика что и в полном переборе, только добаляем не в общий стэк а в s2

                iter++

                if(iter > maxIterations){

                    return {result: false, resStatus: 'выход по числу итераций'}

                }

                let resolv = createResolventa(dis1, dis2)

                if(resolv instanceof Disjunction){

                    newDis.push(resolv)

                }

                if(resolv == false){

                    return {result: true, resStatus: 'доказано, найден ложный дизъюнкт'}

                }

            }

        }

        if(newDis.length == 0){

            return {result: false, resStatus: 'новых не добавилось'}

        }

        S2 = [...S2, ...newDis]

    }

}

## 