JavaScript Algorithm for Quantifier Elimination in Epidemiology

Inda Kreso

Sarajevo School of Science and Technology, Sarajevo, Bosnia and Herzegovina

Corresponding author: Inda Kreso. Sarajevo School of Science and Technology, Sarajevo, Bosnia and Herzegovina. E-mail:indakreso1@gmail.com. ORCID ID: http://www.orcid.org:0000-0002-5556-4669.

doi: 10.5455/aim.2018.26.280-283
ACTA INFORM MED. 2018 DEC 26(4): 280-283
Received: Oct 10, 2018 • Accepted: Nov 26, 2018

ABSTRACT

Introduction: Quantifier Elimination gives us perfect insight into the most basic world of the computer, its origin, its primer functions and it basic operations. Carefully designed and programmed Algorithm for Quantifier Elimination makes the quantifier elimination from the quantified formulas much easier and much more comprehensive **Aim:** This paper explains how Quantifier Elimination algorithm can be used in the field of Biology, or to be more specific, in the field of Epidemiology. **Material and methods:** Exemplary formulas needed for the algorithm are all the formulas from the Mathematical Logic field. JavaScript programming language was used in order to program fast and effective algorithm for Quantifier Elimination. **Results:** Solving the certain problems from the field of Epidemiology using the Quantifier Elimination method, proved to be very successful in the past, because it made possible for the results to be extracted very fast. Doing the exact thing using the newer generation algorithm might be even more effective. **Conclusion:** The most basic concepts of Mathematical Logic can be implemented in order to solve the one of the most important questions in Epidemiology.

Keywords: quantifiers, elimination, algorithm, epidemiology, disease dynamics.

1. INTRODUCTION

The discipline of Mathematical Logic was invented in the first half of the 20th century by a huge number of amazing mathematicians such as Frege, Hilbert, Gödel, Turing, Tarski, Malce and many others (1). The development of this mathematical field is without a doubt one of the highest achievements and successes of science in the 20th century because it presented logical reasoning and computability to high-developed analysis, and eventually led to the creation of computers (2-8).

To be more specific, Mathematical Logic is used for the formalization of the semantics of various programming languages and for verifying the correctness of the programs. Quantifier Elimination is one of the most interesting topics that Mathematical Logic explores (2).

Even though Quantifier Elimination is a method that comes from the world of the computers, its biggest application is actually in the world of Epidemiology. In the world of Epidemiology, one of the most important information is about the dynamics of certain disease (3). Quantifier Elimination method together with the JavaScript algorithm can provide these information very quickly.

2. MATERIAL AND METHODS

The main difficulty encountered in the process of programming the Quantifier Elimination algorithm was actually the limitation of the JavaScript programming language. Since the JavaScript is web-oriented programming language, it does not support some of the functions that can be easily found in the Mathlab or Mathematica programming languages. Unfortunately, some of the functions that are implemented in the Mathematica programming language are not implemented in JavaScript programming language, and that is why very challenging to program this algorithm in JavaScript programming language.

The functions that were not implemented in the JavaScript language are written step by step in order to do the exact same thing as the ones that are already implemented in Mathematica programming language. In order to

implement all the function needed, the extensive math library math.js was used.

The JavaScript version of the algorithm has some constraints. These constraints are the results of the differences in these two programming languages: JavaScript and Mathematica. The JavaScript version of the algorithm works on the predefined range of numbers form -100 to 100. This is range that was set at the beginning and it can be changed to be bigger, but it cannot be infinite as it is the case with the Mathematica version of the algorithm. Since the algorithm is based on the Boolean functions, the final result is TRUE or FALSE. In general, quantifier elimination method tends to decide when a sentence from the theory is true, and then to computer the quantifier free formula (new formula without the quantifiers that is equivalent with the one with the quantifiers). The complete code of the JavaScript algorithm can be found in the Appendix.

3. RESULTS

According to Maric et al. (4) the change of quantitative behavior of parameterized system differential equations that are encountered in the epidemiology, are represented by the rational functions of the parameters. Form the computational point of view, there was a difficulty with solving the following question: if one rational function equal to 1, does that mean that the other rational function is less than zero for all parameters (4). The traditional algebra was not able to solve this question, but if this question is defined as a first order formula, then it can be solved using the quantifier elimination method (5).

The basic problem in the field of epidemiology is to gather the biggest amount of information regarding the dynamics of the certain disease (6). The most important parameter in the epidemiology is the basic reproduction number that is represented by: This number represents average number of secondary infections generated by one case, and automatically the expression that describes the dynamics of the disease is given: if the average number of secondary infections is smaller than 1 (), then the disease will most certainly die out, and in the other case, when the average number of secondary infections is bigger than 1 (), then the epidemic will be expected. The main problem according to (7) is to represent a formula for the total population that depends on. In order to that, there has to be a epidemic model of a certain disease. The population is usually divided into three groups: susceptible infected and treated infected All the parameters that influence all of three groups can be represented by the non-linear differential equations. After the range of mathematical operations, this first formula can be formulated as a quantified formula, as it was suggested at the beginning of this section. According to Chauvin C. et al. (4) Quantifier Elimination method is used in order to solve this problem, and it proves to be really quickly, but with the JavaScript algorithm is tends to be even more faster than with the previous method.

The epidemic model that was descried earlier can be expressed using the non-linear differential equations of the form:

$$\frac{dX_i}{dt} = \mu N_0 \gamma_1 - (\rho_1 \lambda + 1) X_i$$

$$\frac{dY_i}{dt} = \rho_1 \lambda X_i - (v + t + 1) Y_i$$

$$\frac{dV_i}{dt} = \tau Y - (\delta + \mu) V_i$$

Legend: X_i-susceptibles , Y_(i)– infected, V_i -treated infected μ-natural mortality rate, γ_1 and γ_2- the portion of the whole population going into the high and low part of the population, ρ_1 λ-force of infection, δ-disease induced mortality rate, N_0-stable, population in absence of infection, τ-rate of getting treated

The complete mathematical calculations can be found in the article under the name: "An application of Quantifier Elimination to Mathematical Biology" by Corinne Chauvin, Myriam Mueller and Andreas Weber (4). The complete mathematical calculations will not be included in this paper, because the focus is not mainly on mathematical point of view, rather on the computer science point of view. The quantified formula of the result of solving the non-linear system of differential equations is the following one:

$$\forall c \in (0,1), \forall t > 0, \forall h, \forall d > 0, \forall \gamma_2 < \tfrac{1}{2}, \forall v > 0, \forall r_1 > 0, \forall r_2 > 0, \forall p_1 : r_1 < r_2 \wedge d < v \Rightarrow (R_\tau = 1 \wedge P(p_1) = 0 \Rightarrow p_1 \leq 0)$$

After the formula without quantifiers is obtained, the JavaScript algorithm can be implemented directly on the formula in order to get the needed result.

4. DISCUSSION

In the filed of a Predicate logic there exist two quantifiers: universal and existential (6). A universal quantification is a type of the quantifier that is interpreted as "given any" or "for all". It means that every member of the domain can satisfy a propositional function. Furthermore, it also means that a predicate within the scope of a universal quantifier is true for every value of a predicate variable (7).

It is denoted by, which is combined with a predicate variable in order to form a universal quantifier: .

Example:

The following example is the example for the usage of the universal quantifier. Suppose that is given: For all natural numbers n, 2 · n = n + n. This first part: for all natural numbers is actually universal quantifier.

$$2 \cdot 0 = 0 + 0 \; and \; 2 \cdot 1 = 1 + 1 \; and \; 2 \cdot 2 = 2 + 2 \ldots$$

It can be seen that for every number the rule is the same (11).

The second quantifier is called an existential quantifier (9, 10). In predicate logic, an existential quantification is a type of quantifier, a logical constant that is interpreted as "there exists," "there is at least one". It means that at least one member of the domain can satisfy a propositional function. It proposes that a predicate within the scope of an existential quantifier is true for at least one value of a variable (11, 12). It is denoted by the symbol , which is combined with a predicate variable in order to form an existential quantifier: Quantifier elimination is an idea of simplification that is widely used in many fields such as: mathematical logic, theoretical com-

puter science and many more (9). Quantifier elimination is the main technique that is used to eliminate quantifiers from the formulas (10). Formulas can be classified by the amount of the quantifiers they have, and the formulas that have less quantifiers are considered to be simpler ones. Suppose it is given language L and a set A of the formulas in L. Set A allows quantifier elimination in the formula F of the language L if there exists a formula without quantifies such that (2). In other words quantifier elimination can be performed if for every formula there exists another formula that has no quantifiers and it is equivalent to the formula with quantifier (2).

5. CONCLUSION

Since the Quantifier Elimination algorithm has proven to be one of the basic tools that can be used in the field of Mathematical Logic, it is not surprising that this same tool found its broad application in the other sciences such as Computer Science and Biology. This paper presents the great overview how basic concepts of Mathematical Logic can be turned into a highly comprehensive and web-oriented algorithm programmed in one of the world's most popular programming language JavaScript, and then be used in order to solve the one of the most important problems in Epidemiology (9).

- **Author's contribution:** Author was included in all steps of preparation of this article and made final proof reading before printing.
- **Financial support and sponsorship:** None.
- **Conflict of interest:** There are no conflict of interest.

REFERENCES

1. Kostic M. Elementi teorije sistema i informacija. Naucna knjiga. Beograd, 1990: 46-56.
2. Dedic S. Osnovi nauke o upravljanju. Svjetlost. Sarajevo, 1986: 227-231.
3. Masic I, Ridjanovic Z Sistem i komunikacija. U: Medicinska informatika. Avicena. Sarajevo, 1994: 85-140.
4. Chauvin C, Muller M, Weber A. An Application of Quantifier Elimination to Mathematical Biology. 287-296. Retrieved: https://www.researchgate.net/profile/Andreas_Weber4, November 15th, 2018
5. Kern J, Petrovecki M. Medicnsko odlucivanje. U: Kern J, Petrovecki M (Eds.). Medicinska informatika. Medicinska naklada. Zagreb. 2009: 179-196.
6. Masic I, Ridjanovic Z. Hardver i softver. U: Masic I, Ridjanovic Z. Medicinska informatika. Avicena. Sarajevo. 1999: 195-266.
7. Mašić I, Riđanović Z. Medicinska informatika, knjiga II, Aplikativna medicinska informatika, Avicena, Sarajevo, 1999.
8. Masic I, Ridjanovic Z, Pandza H, Masic I. Medical Informatics. Avicena. Sarajevo, 2010.
9. Shortliffe EH, Perreault LE, Wiederhold G, Fagan LM. (Eds) Medical Informatics. Computer Applications in Health Care and Biomedicine. Springer. New York, Berlin, Heidelber. 2001: 327-358
10. Chapman SJ. Multiscale mathematical modelling in medicine and biology. OCIAM. Mathematical Institute, 24-29 St Giles, Oxford OX1 3LB, 2015.
11. Bender EA. An Introduction to Mathematical Modelling, New York: Dover, 2000.
12. Tan JKH. Health Management Information Systems. Theories, Methods and Applications. Aspen Publication. 1995: 65-125

APPENDIX

```
var RANGE_ERROR = "CHECK YOUR RANGE. SHOULD BE FROM -100
TO 100.";
var NO_FUNCTION_ERROR = "NO FUNCTION ERROR. PLEASE CHECK
FUNCTION.";

var set = new Array(201);

for(var i = 0; i <= 200; i++) {
    set[i]=i-100;
}
console.log(math.sqrt(-4));

// create a parser
var parser = math.parser();


function go() {

var formulaStartString = document.
getElementById("formula").value;

var operation;
var res = solve(formulaStartString);
    setResult(res);

}

function solve(formulaStartString) {
var expressions = formulaStartString.
substring(formulaStartString.
indexOf("(")+1,formulaStartString.lastIndexOf(")"));
var expressionsArray;

if(expressions.indexOf("),") != -1) {
        expressionsArray = expressions.split("),");
        expressionsArray[0] = expressionsArray[0] + ')';
    }
else expressionsArray = expressions.split(",");

    expressionsArray = expressionsArray.map(element => {
return element.trim();
    });

if(!(formulaStartString.startsWith("Exists") ||
formulaStartString.startsWith("ForAll")
        || formulaStartString.startsWith("AND") ||
formulaStartString.startsWith("OR")
        || formulaStartString.startsWith("NOT") ||
formulaStartString.startsWith("Implies")
        || formulaStartString.startsWith("Equivalent")))
{
        document.getElementById("result").innerHTML =
NO_FUNCTION_ERROR;
return;
    }
if(formulaStartString.startsWith("Exists")) {
return exists(expressionsArray[0], expressionsArray[1]);
//setResult(result);
    }
elseif(formulaStartString.startsWith("ForAll")) {
return forAll(expressionsArray[0], expressionsArray[1]);
//setResult(result);
    }
elseif(formulaStartString.startsWith("AND")) {
return and(expressionsArray[0], expressionsArray[1]);
    }
elseif(formulaStartString.startsWith("OR")) {
return or(expressionsArray[0], expressionsArray[1]);
    }
elseif(formulaStartString.startsWith("NOT")) {
return not(expressionsArray[0]);
    }
elseif(formulaStartString.startsWith("Implies")) {
return implies(expressionsArray[0], expressionsArray[1]);
    }
elseif(formulaStartString.startsWith("Equivalent")) {
return equivalent(expressionsArray[0],
expressionsArray[1]);
    }
else document.getElementById("result").innerHTML = NO_
FUNCTION_ERROR;
}

function setResult(result) {
    document.getElementById("result").innerHTML = result;
}

function exists(exp1, exp2) {
var exp1Result, exp2Result;
if(exp1.startsWith("Exists") || exp1.startsWith("ForAll")
        || exp1.startsWith("AND") || exp1.
startsWith("OR")
        || exp1.startsWith("NOT") || exp1.
startsWith("Implies")
        || exp1.startsWith("Equivalent")) {
            exp1Result = solve(exp1);
        }
elseif(exp1.toString() == 'x') {
        exp1Result = false;
    }
```

```
else {
        exp1Result = evaluateExp(exp1, true);
    }

if(exp2.startsWith("Exists") || exp2.startsWith("ForAll")
        || exp2.startsWith("AND") || exp2.
startsWith("OR")
        || exp2.startsWith("NOT") || exp2.
startsWith("Implies")
        || exp2.startsWith("Equivalent")) {
            exp2Result = solve(exp2);
    }
elseif(exp2.toString() == 'x') {
        exp2Result = false;
    }
else {
        exp2Result = evaluateExp(exp2, true);
    }
return exp1Result || exp2Result;
}

function forAll(exp1, exp2) {
if(exp1.toString() == 'x')
return evaluateExp(exp2, false);
elsereturn evaluateExp(exp1, false) && evaluateExp(exp2,
false);
}

function and(exp1, exp2) {
var exp1Result, exp2Result;
if(exp1.startsWith("Exists") || exp1.startsWith("ForAll")
        || exp1.startsWith("AND") || exp1.
startsWith("OR")
        || exp1.startsWith("NOT") || exp1.
startsWith("Implies")
        || exp1.startsWith("Equivalent")) {
            exp1Result = solve(exp1);
        }
elseif(exp1.toString() == 'x') {
        exp1Result = true;
    }
else {
        exp1Result = evaluateExp(exp1, true);
    }

if(exp2.startsWith("Exists") || exp2.startsWith("ForAll")
        || exp2.startsWith("AND") || exp2.
startsWith("OR")
        || exp2.startsWith("NOT") || exp2.
startsWith("Implies")
        || exp2.startsWith("Equivalent")) {
            exp2Result = solve(exp2);
    }
elseif(exp2.toString() == 'x') {
        exp2Result = true;
    }
else {
        exp2Result = evaluateExp(exp2, true);
    }
return exp1Result && exp2Result;
}

function or(exp1, exp2) {
var exp1Result, exp2Result;
if(exp1.startsWith("Exists") || exp1.startsWith("ForAll")
        || exp1.startsWith("AND") || exp1.
startsWith("OR")
        || exp1.startsWith("NOT") || exp1.
startsWith("Implies")
        || exp1.startsWith("Equivalent")) {
            exp1Result = solve(exp1);
        }
elseif(exp1.toString() == 'x') {
        exp1Result = false;
    }
else {
        exp1Result = evaluateExp(exp1, true);
    }

if(exp2.startsWith("Exists") || exp2.startsWith("ForAll")
        || exp2.startsWith("AND") || exp2.
startsWith("OR")
        || exp2.startsWith("NOT") || exp2.
startsWith("Implies")
        || exp2.startsWith("Equivalent")) {
            exp2Result = solve(exp2);
    }
elseif(exp2.toString() == 'x') {
        exp2Result = false;
    }
else {
        exp2Result = evaluateExp(exp2, true);
    }
return exp1Result || exp2Result;
}

function not(exp1) {
var exp1Result;
if(exp1.startsWith("Exists") || exp1.startsWith("ForAll")
        || exp1.startsWith("AND") || exp1.
startsWith("OR")
        || exp1.startsWith("NOT") || exp1.
startsWith("Implies")
        || exp1.startsWith("Equivalent")) {
            exp1Result = solve(exp1);
        }
elseif(exp1.toString() == 'x') {
        exp1Result = true;
    }
else {
        exp1Result = evaluateExp(exp1, true);
    }

return !exp1Result;
}

function implies(exp1, exp2) {
var exp1Result, exp2Result;
if(exp1.startsWith("Exists") || exp1.startsWith("ForAll")
        || exp1.startsWith("AND") || exp1.
startsWith("OR")
        || exp1.startsWith("NOT") || exp1.
startsWith("Implies")
        || exp1.startsWith("Equivalent")) {
            exp1Result = solve(exp1);
        }
elseif(exp1.toString() == 'x') {
        exp1Result = false;
    }
else {
        exp1Result = evaluateExp(exp1, true);
    }

if(exp2.startsWith("Exists") || exp2.startsWith("ForAll")
        || exp2.startsWith("AND") || exp2.
startsWith("OR")
        || exp2.startsWith("NOT") || exp2.
startsWith("Implies")
        || exp2.startsWith("Equivalent")) {
            exp2Result = solve(exp2);
    }
elseif(exp2.toString() == 'x') {
        exp2Result = false;
    }
else {
        exp2Result = evaluateExp(exp2, true);
    }
return (exp1Result === true&& exp2Result === true)
|| (exp1Result === false&& exp2Result === true) ||
(exp1Result === false&& exp2Result === false);
}

function equivalent(exp1, exp2) {
var exp1Result, exp2Result;
if(exp1.startsWith("Exists") || exp1.startsWith("ForAll")
        || exp1.startsWith("AND") || exp1.
startsWith("OR")
        || exp1.startsWith("NOT") || exp1.
startsWith("Implies")
        || exp1.startsWith("Equivalent")) {
            exp1Result = solve(exp1);
        }
elseif(exp1.toString() == 'x') {
        exp1Result = false;
    }
else {
        exp1Result = evaluateExp(exp1, true);
    }

if(exp2.startsWith("Exists") || exp2.startsWith("ForAll")
        || exp2.startsWith("AND") || exp2.
startsWith("OR")
        || exp2.startsWith("NOT") || exp2.
startsWith("Implies")
        || exp2.startsWith("Equivalent")) {
            exp2Result = solve(exp2);
    }
elseif(exp2.toString() == 'x') {
        exp2Result = false;
    }
else {
        exp2Result = evaluateExp(exp2, true);
    }
return exp1Result === exp2Result;
}

function evaluateExp(exp, satisfy) {

for(var i = 0; i <201; i++)
    {
        console.log(set[i]);
var exp1 = exp.replace("x", set[i] >= 0 ? set[i] : '(' +
set[i] + ')');

        console.log("eval",exp);

        console.log(math.eval(exp1));

if(math.eval(exp1) == satisfy)
return satisfy;
    }

return !satisfy;}
```