Animation of n-queens problem in JavaScript

Koen Logmann & Jessica Roth

Study report

T_3200

Course of Studies: Applied Computer Science

Department of Computer Science

Baden-Wuerttemberg Cooperative State University Mannheim

March 7, 2019

Tutors

Prof. Dr. Karl Stroetmann, DHBW Mannheim

Logmann, Koen & Roth, Jessica:
Animation of n-queens problem in JavaScript / Koen Logmann & Jessica Roth. –
Bachelor Thesis, Mannheim: Baden-Wuerttemberg Cooperative State University Mannheim,
2019. 14 pages.

Logmann, Koen & Roth, Jessica:
Animation des N-Damen Problems in JavaScript / Koen Logmann & Jessica Roth. –
Bachelor-Thesis, Mannheim: DHBW Mannheim, 2019. 14 Seiten.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, March 7, 2019

Koen Logmann & Jessica Roth

Abstract

Animation of n-queens problem in JavaScript

Todo

Animation des N-Damen Problems in JavaScript

TODO

Contents

Chapter 1

Introduction

Figure 1.1: DHBW-Logo [**lin1973**]

Chapter 2

Scientific Basics

The aim of this work is to visualize the Davis Putman algorithm that solves the so-called n queens problem.
Therefore a general understanding of this algorithm and the mathematical problem has to be created.
For this reason, this chapter summarizes this fundamental knowledge in order to create a basis for further development. Among other things, the declaration of the mathematical problem plays a role here, so that it can be solved by the Davis Putman algorithm.

2.1 Davis Putnam algorithm

The Davis Putman algorithm is a method for calculating a solution of logical clause sets. With very small clause sets this can be easily determined, as can be seen in the following two examples.

$$K_1 = \{\ \{r\},\ \{\neg s\},\ \{t\},\ \{\neg u\},\ \{\neg v\}\ \}$$

K_1 can also be written as a statement logic formula.

$$r \wedge \neg s \wedge t \wedge \neg u \wedge \neg v.$$

It is recognizable that this formula is solvable by r and t "true" and s, u and v having the value "false". As a counter example K_2 is to be considered.

$$K_2 = \{\ \{r\},\ \{\},\ \{t\}\ \}$$

K_2 can also be written as a statement logic formula.

$$r \wedge \bot \wedge t.$$

An empty set means a falsum in the statement logic, making K_2 impossible to fulfill. With very large clause sets it is usually no longer visible at first glance, so that algorithms like this are used. But to be able to set a step lower, two definitions have to be introduced first. [1]

Unit clause A clause C is a unit clause if it consists of only one literal, i.e. a statement variable.

trivial clause sets A trivial set of clauses can only occur if one of the two cases occurs.

1. K contains the empty clause and is therefore unfulfillable
2. The unit clauses always contain different statement variables, so that only the clause $\{p\}$ or $\{\neg p\}$ can occur. If this is the case, a solution for the clause set can be determined.

In order for one of these two cases to occur, the clause sets must be simplified with the help of the following three options so that they consist only of unit clauses.

1. Cut Rule
2. Subsumption
3. Case Differentiation

2.1.1 Simplification with Cut Rule

To simplify the set of clauses K the cut rule can be used. A usual application of the cut rule is.

$$\frac{C_1 \cup \{l\} \qquad C_2 \cup \{\neg l\}}{C_1 \cup C_2}$$

In this case the result clause will usually have more literals than $C_1 \cup \{l\}$ or $C_2 \cup \{\neg l\}$ alone. Because if the clause $C_1 \cup \{l\}$ contains $m+1$ literals and $C_1 \cup \{l\}$ contains $n+1$ literals, then the union $C_1 \cup C_2$ can have upto $m+n$ literals in total. In general $m+n$ is bigger than $m+1$ or $n+1$. The clause can be smaller than previously if both sets contain the same literals, but it is only granted to be smaller if $m = 0 \vee n = 0$. This is the case for unit clauses. Therefore we will only allow the use of the cut rule

if one of the clauses is a unit clause. These cuts will be called unit cuts. To be able to do those unit cuts with a unit clause $\{\ l\ \}$ on all clauses of a set of clauses K. We define the function

$$unitCut : 2^K \times L \to 2^K$$

so that for a set of clauses K and a literal l the function $unitCut(K,\ l)$ will simplify the set of clauses as much as possible by using unit cuts:

$$unitCut(K,\ l) = \{\ C \setminus \{\ \neg l\ \} \mid C \in K\ \}$$

The result set of clauses will have the same amount of clauses as K. Only the clauses $C \in K$ where $\neg l \in C$ were affected and got that element removed. [2]

2.1.2 Simplification with Subsumption

For further simplification of the set of clauses we will use subsumption. The idea is simple and can be shown with the following example:

$$K = \{\ \{\ l,\ p,\ \neg q\ \},\ \{\ l\ \}\ \} \cup M$$

It is clear that the clause $\{\ l\ \}$ implifies the clause $\{\ l,\ p,\ \neg q\ \}$, because if $\{\ l\ \}$ is fulfill, then $\{\ l,\ p,\ \neg q\ \}$ will be fulfill too. That's because

$$\models l \to l \vee p \vee \neg q$$

is given. With that in mind we can say that we can subsume a clause C with a unit clause U if $U \subseteq C$. That means if we have a set of clauses K with $C \in K \wedge U \in K$ and U can subsume C, we can remove the clause C from K to reduce its size. We define a function

$$subsume : 2^K \times L \to 2^K$$

that we will give it a set of clauses K and a literal l, so that it will simplify K by subsuming K with the unit clause $\{\ l\ \}$ and keep the unit clause itself:

$$subsume(K,\ l) = \{\ C \in K \mid l \notin C\} \cup \{\{\ l\ \}\}$$

We have to add the unit clause to the set of clauses, because $l \in U$ is the case. Resulting in U being removed from K at first too. [2]

2.1.3 Simplification with Case Differentiation

The following sentence forms the basis for the principle of case differentiation.

Theorem The clause set K can be fulfilled if the clause $K \cup \{\{p\}\}$ or $K \cup \{\{\neg p\}\}$ can be fulfilled.

For simplification, a statement variable p is selected at the beginning, which occurs in the clause set. Then the two clause sets mentioned above are formed and an attempt is made to find a solution for one of them. If this attempt is successful, the result is automatically the solution of K. If none is found, K is unsolvable.

2.1.4 Vorgehen des Algorithmus

The knowledge base that was previously created now makes it possible to sketch the procedure of the Davis Putman algorithm. With the help of the intersection rule and the subsumption, the clause set K is simplified as far as possible. If already after this step K is trivial, the procedure is finished. Otherwise a statement logical variable p is selected, which occurs in K. Then a recursive attempt is made to solve the clause set $K \cup \{\{p\}\}$ in order to find a solution for K. If no solution was found here either, the same is tried with the negated p. If this attempt also fails, K is unsolvable.[1], [2]

2.2 N Queens Problem

The n queen problem is the generalized mathematical problem related to a chessboard that consists of $n \times n$ squares. A special example would be the 8 queens problem, which is related to the standardized chessboard. In general, the problem is to place n queens on an $n \times n$ chessboard so that none would be obstructed in their turn. A queen in a normal game of chess can move diagonally, vertically and horizontally. This move pattern can be seen in Figure 2.1. In summary, this means that there is only one queen allowed on her vertical, horizontal and diagonal line at a time so that they do not interfere with each other. In this problem it is assumed that any queen can attack any other queen and the field colors are ignored. This problem can be solved by several algorithms such as the Davis Putman algorithm. [3], [4], [5, pp. 146 sq.], [2]

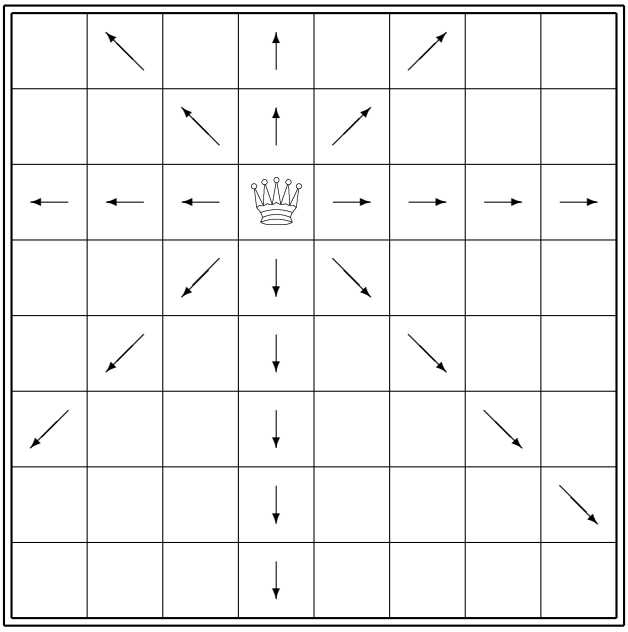

Figure 2.1: Das 8-Damen-Problem [6]

Chapter 3

Technical Basics

Chapter 4

Implementation

Chapter 5

Prospect

List of Abbreviations

List of Tables

List of Figures

Bibliography

[1] H. Zhang and M. E. Stickel, *Implementing the davis–putnam method*, https://www.math.ucdavis.edu/~deloera/TEACHING/MATH165/davisputnam.pdf, Accessed on 2018-10-23, Oct. 2000.

[2] K. Stroetman, *Theoretical computer science i : Logic*, Feb. 16, 2019. [Online]. Available: https://github.com/karlstroetmann/Logik/blob/master/Lecture-Notes-Python/logic.pdf (visited on 03/06/2019).

[3] J. Bell and B. Stevens, "A survey of known results and research areas for queens", vol. 309, pp. 1–31, Jan. 6, 2009. DOI: 10.1016/j.disc.2007.12.043. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0012365X07010394?via%3Dihub.

[4] J. J. Watkins, *Across the Board - The Mathematics of Chessboard Problems*. Princeton University Press, Jul. 3, 2012, 272 pp. [Online]. Available: https://www.ebook.de/de/product/18353808/john_j_watkins_across_the_board_the_mathematics_of_chessboard_problems.html.

[5] S. P. Nudelman, "The modular n-queens problem in higher dimensions", vol. 146, pp. 159–167, 1995. DOI: 10.1016/0012-365x(94)00161-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0012365X94001615?via%3Dihub.

[6] K. Stroetman, *Aussagenlogik*, Feb. 16, 2019. [Online]. Available: https://github.com/karlstroetmann/Logik/blob/master/Lecture-Notes-Python/aussagenlogik.tex (visited on 03/06/2019).