K Modal Logic Approach to Maze Solving

Dinko Osmanković
Faculty of Electrical Engineering
University of Sarajevo
71000 Sarajevo, Bosnia&Herzegovina
Email: dinko.osmankovic@etf.unsa.ba

Aleksandar Aćimović
Faculty of Electrical Engineering
University of Sarajevo
71000 Sarajevo, Bosnia&Herzegovina
Email: aacimovic1@etf.unsa.ba

***Abstract*—This paper presents a modal logic approach to solving maze problem. Traditionally, this problem is solved using graph traversal algorithms, ranging from backtracking and A* to monte–carlo tree search approaches. In this case, we present a novel approach to describing a backtracking algorithm using K modal logic. Modal logic is a powerful way of representing statements that involve some linguistic modalities, e.g. "necessary", "possible", "was", "will be", "permitted", "forbidden", etc. This gives new ways to model typical decision making problems. For this purpose, we developed a novel framework called LogicToolbox. We employ this tool in modelling and solving maze problem as an example of a decision making process. We chose to *translate* a well–known backtracking algorithm to present the expressiveness of the language of modal logic, while proving that solving backtracking as a modal logic problem is equivalent to the backtracking algorithm itself.**

I. Introduction

Maze problem is very simple to state. An agent, who is limited to perceive its only close surrounding, is required to find the way out of a map with walls staring from a certain position. Although simple, the applications range from robotics [1], [2], urban traffic planning [3] to psychology [4].

However, the performance of finding the exact solution to the maze problem is decreasing significantly with the size of the maze [5]. Several heuristic approaches have been presented, e.g. A* search [6], biochemistry-inspired algorithms [7]–[9], machine learning approaches [10]–[12], etc.

Maze can as well be seen as a decision making problem [13]. In this sense, decision making and decision space exploration can be viewed as a formal logic reasoning [14]. For the purpose of expressivity of formal language, modal logic was developed to extend the notion of decision events in terms of the linguistic modalities that sometimes appear along decision events. Several types of modalities can be attributed to events leading to different types of modal logic. These include:

Modal Logic	$\Box$	It is necessary that ...
	$\Diamond$	It is possible that ...
Deontic Logic	O	It is obligatory that ...
	P	It is permitted that ...
	F	It is forbidden that ...
Temporal Logic	G	It will always be the case that ...
	F	It will be the case that ...
	H	It has always been the case that ...
	P	It was the case that ...
Doxastic Logic		x believes that ...

These modal operators are added to the propositional logic in order to increase expressivity of a logical formula. With this in mind, decision making process can be more refined to include modalities to express belief, time dependencies and object dependencies, etc. In [15] authors apply these principles to develop multi-agent game tree search for AI game players in games of Hex and Chess. Prisoner's Dilemma, Battle of the Sexes and Matching Pennies are 2×2 games (know in game theory) that are modelled and solved using modal logic as presented in [16].

The rest of the paper is organised as follows. In Section II, the methodology behind our proposed approach including LogicToolbox framework for evaluating modal formulas and the backtracking solution to the maze problem in the language of modal logic is presented. In section III the results of numerical simulations are given. Finally, last section concludes the paper.

II. Methodology

In this section we present the building blocks of maze solving algorithm devised as a propositional modal logic formula. If this formula is satisfiable, it means that there is a solution to the maze problem. Moreover, if there is only one set of values for formula variables for which the formula is satisfiable, then there is exactly one solution to the maze problem. Note that proper maze problem itself has exactly one solution.

A. Kripke semantics

Kripke semantics is a way to represent the context in which the truth–value of a modal formula is determined [17], [18]. This semantics is given as a triple $(W, R, \vDash)$, where W is a set of *worlds* (possibly empty), R is a binary relation on W called *accessibility relation* and $\vDash$ is a *satisfaction relation* between worlds of W and modal formula F such that for $w \in W$, $w \vDash F$ means that formula F is satisfiable in world w.

A formula F is valid or tautology in $(W, R, \vDash)$ if and only if $w \vDash F, \forall w \in W$. If $\exists w \in W$ such that $w \vDash F$, then we say that formula F is satisfiable in $(W, R, \vDash)$, otherwise, the formula F is a contradiction. Formula $\Box F$ is valid in $w \in W$ if and only if formula F is valid in all worlds in W accessible (by relation R) from world w. Formula $\Diamond F$ is valid in $w \in W$ if there is at least one world $v \in W$ accessible from w in which formula F is valid.

978-1-5386-4640-3/18/$31.00 ©2018 IEEE

Depeding on the properties of accessibility relation R, several modal logic systems arise [18]:

- **K** - no restrictions,
- **D** - R is serial relation,
- **T** - R is reflexive relation,
- **B** - R is reflexive and symmetric,
- **S4** - R is reflexive and transitive,
- **S5** - R is reflexive and Euclidean.

For the maze problem, **K** modal logic is an adequate system required for its modelling. In general, having no restrictions on accessibility relation gives more flexibility in designing Kripke model of the particular problem to be solved.

B. LogicToolbox - Framework for Propositional Modal Logic

LogicToolbox is an open source framework, written in C++, for determining the truth–value of a modal formula F in all possible contexts, i.e. worlds. Framework provides the following object types:

- Variable,
- World,
- Universe (Kripke model),
- Expression (modal formula) along with classes for expression parsing.

The relationship between these classes are represented with class diagram in Fig. 3.

The framework is designed and implemented in order to solve typical Kripke models. For example, let the formula

$$F = \Diamond(x \vee y) \rightarrow \Box(x \wedge y) \tag{1}$$

be given. Obviously, there are four possible worlds in which x and y can have truth–values, i.e. w_1 where $(x, y) = (\top, \top)$, w_2 where $(x, y) = (\bot, \top)$, w_3 where $(x, y) = (\top, \bot)$, and w_4 where $(x, y) = (\bot, \bot)$. Framework determines truth–value of formula F for all the worlds in Kripke model. The example accessibility relation is represented as graph as shown in Fig. 2. Note that one can design accessibility relation R in a different way, hence the even more expressivity of modal logic.

As shown, formula F, given in (1), is true in worlds w_2, w_3 and w_4 and false in w_1. Therefore, formula F is satisfiable in the given Kripke model.

Modal formulae are parsed from string using Shunting Yard algorithm [19] to build an expression tree. Satisfiability and validity tests are performed on such a model using depth-first search approach [20], [21].

LogicToolbox is an open source framework, available under MIT Licence, and available at https://github.com/dinkoosmankovic/LogicToolbox.

C. Maze Problem as Modal Logic Formula

Now we present the modal logic – based solution to the maze solving problem. Modal logic is a powerful way to represent linguistic modalities and to model a typical decision making routine. Therefore, maze solving can be seen as a decision making problem solved employing modal logic formulae.

First, the maze can be represented as a decision graph. Each state is represented as a graph node, meaning that each position in a maze is a node in such a graph. An agent can go in four directions: UP, DOWN, LEFT and RIGHT. Each of these actions are represented as directed graph edges going from one state to the other state. Two special states are starting and goal positions. This is shown in a Fig. 3.

In a typical backtracking algorithm, an agent would start from a state x_s and then check all the unvisited neighbours. It will choose one, and put all the others on stack. This procedure is repeated until the state x_g is reached. The whole algorithms is presented in Algorithm 1, where G is a decision graph of a maze.

Algorithm 1 Pseudocode of the backtracking algorithm for Maze Solving

```
1: function BACKTRACKING(G,x)
2:     S = {}
3:     S.push(x)
4:     while S in not empty do
5:         x = S.pop()
6:         if x is not visited then
7:             x.visited = true
8:             for ∀y ∈ G such that there is an edge from x to y do
9:                 S.push(y)
10:                if y = x_g then return true
11:                end if
12:            end for
13:        end if
14:    end while
15: return false
16: end function
```

Analysing the pseudocode of the backtracking algorithm for maze solving, one can notice the two **if-then** constructs that are typically described with conditional logical connective (or implication). This means that the first **if-then** construct, together with **for-loop** can be expressed as:

$$V(y) := V(x) \wedge \exists y(y \in G \wedge (x, y) \in E(G) \wedge \neg V(y)) \tag{2}$$

This formula represents the *visited* truth value for some state y. It is iteratively evaluated in each state in the maze, meaning that if $V(x_g)$ is true it yields the solution to the maze problem. $V(x)$ is true if node x is visited, while $E(G)$ represents the set of edges of graph G.

However, this formula can be easily translated to the K modal logic framework. In this case, we evaluate the formula:

$$I(x_i) = x \rightarrow \Diamond p_x \tag{3}$$

where x a *visited* truth value of a current state, while $\Diamond p_x$ represent the truth value of the notion that state x is *possibly* a state in a path towards the x_g. This formula is then iteratively evaluated at each state. Now, the problem is the evaluation of $\Diamond p_x$. We propose the solution such that the goal state x_g is given $\Diamond p_{x_g} = \top$. This is represented in Fig. 4.

At this point, we can evaluate $\Diamond p_x$ for each state x going backwards from x_g. This will propagate to the evaluation of

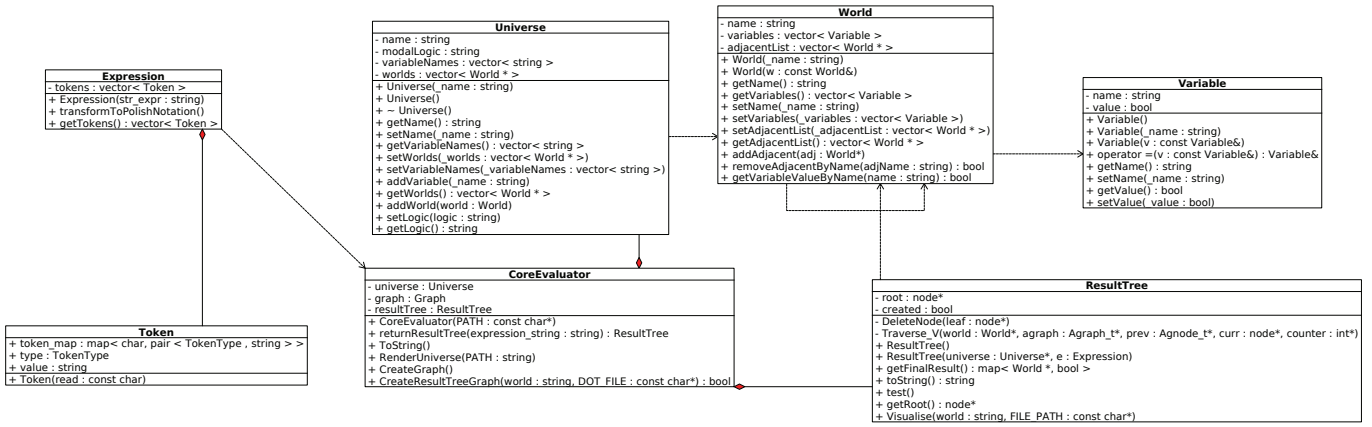

Fig. 1. Class diagram of LogicToolbox

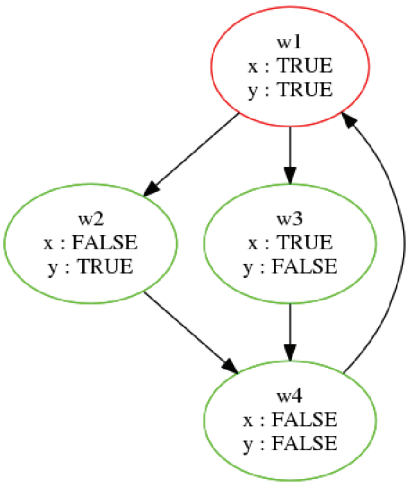

Fig. 2. Visualization of Kripke model where relation R is represented as digraph

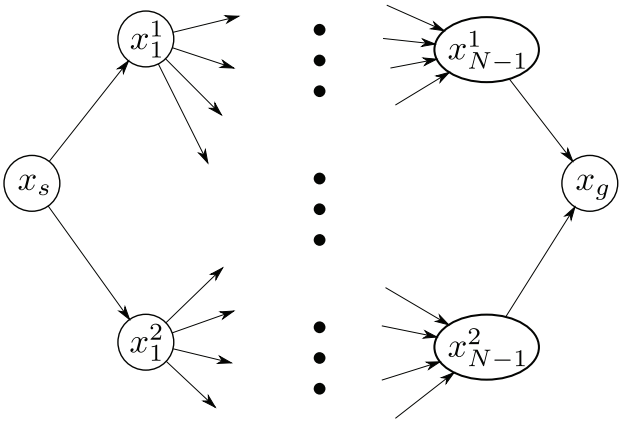

Fig. 3. Graph model of a maze solving problem; State x_s is a starting position, while x_g is a goal position

$\Diamond p_{x_s}$. The example evaluation for the maze problem would go like this:

$$
\begin{aligned}
I = x &\rightarrow \Diamond p_x \\
p_x = x_1 &\rightarrow \Diamond p_{x_1} \\
&\dots \\
p_{x_{N-2}} = x_{N-1} &\rightarrow \Diamond p_{x_{N-1}} \\
p_{x_{N-1}} = x_g &\rightarrow \Diamond p_{x_g}
\end{aligned} \tag{4}
$$

which yields modal formula:

$$I = x \rightarrow (x_1 \rightarrow (\dots \rightarrow \Diamond x_g)\dots) \tag{5}$$

Note that meaning of the construct $\Diamond p_x = \top$ is that there is at least one contextual world, reachable from the current world in which $p_x = \top$. This can be reformulated as there is a neighbouring state the agent can visit that will eventually lead it to the state x_g.

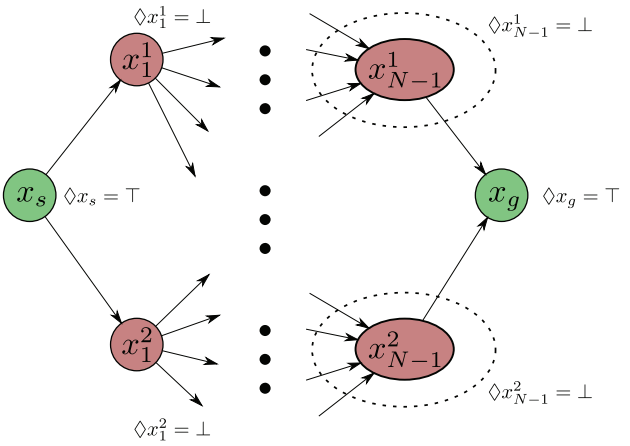

Fig. 4. Evaluating whether the state is possibly on the path leading to x_g

Theorem 1: Evaluation of modal formula given in (5) is equivalent to backtracking algorithm.

Proof: First, let the maze problem be solvable and has only one solution. From equations in (4) it is obvious that modal formula in (5) can be evaluated recursively. Now, if

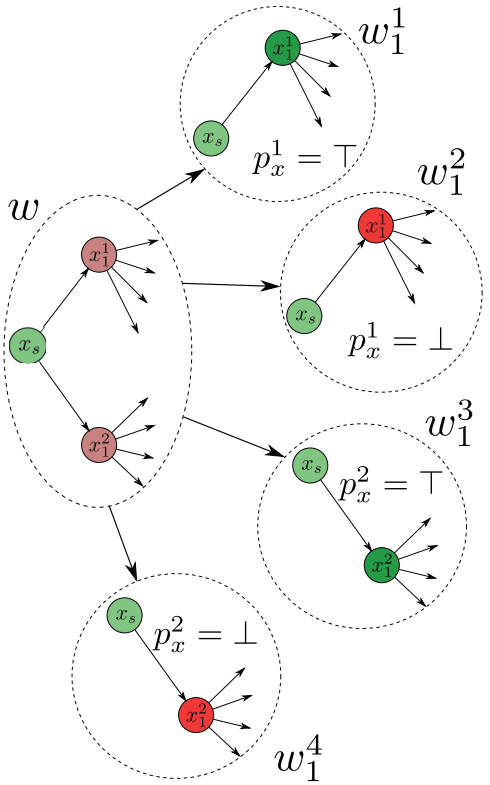

Fig. 5. Graphical representation of the evaluation of the modal formula in the first line in (4)

an agent starts from $p_{x_{N-1}} = \neg x_g \rightarrow \Diamond p_{x_g}$, while setting $\Diamond p_{x_g} = \top$ as an exit strategy. On the other hand, the first step in the recursive evaluation, i.e. $I = x \rightarrow \Diamond p_x$ can be visualised graphically as presented in Fig. 5.

From the current contextual world w in which the starting point is only visited (hence marked with green), two worlds are reachable - one in which $p_x = \top$ and one in which $p_x = \bot$. This means that in world w_1^1, the path will continue along the (x_s, x_1^1) edge, and in world w_1^2 all p_x are then false. Similarly, the same argument can be constructed for other neighbours of x_s yielding four possible contextual worlds reachable from world w. Since, for at least one neighbour of x_s the truth value of $p_x = \top$ i.e. from the assumption that maze is solvable, this evaluation becomes equivalent to the greedy evaluation (for all neighbours of x_s) of $\exists y(y \in G \wedge (x, y) \in E(G))$ from (2). This concludes the proof of the theorem. Note that in general, since there are four neighbours of each graph node (except for start and goal positions), there can be 8 possible worlds accessible from the current world.

■

In the next section, the results and the performance analysis of the proposed approach to maze solving problem are presented.

III. Results

In this secton we present the results obtained for various maze sizes. For this purpose, we defined mazes with sizes

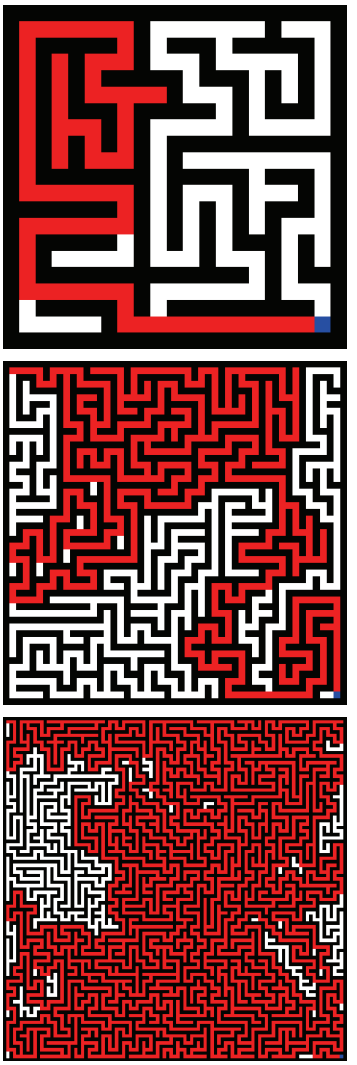

Fig. 6. Maze solutions obtained with modal logic approach to backtracking algortihm; from top to bottom are mazes of size 21x21, 51x51, and 101x101; starting position is in the top left corner, while goal position is represented with blue box in the bottom right corner

11x11, 21x21, 51x51, 101x101, and 201x201 as scenearios for analysis. Both pure backtracking algortihm as well as our proposed modal logic version of backtracking algorithm are discussed.

The solution to maze problems with various sizes is presented in Fig. 6.

For the performance analysis, the algorithms are run 100 times and the average time of execution is calculated. This is represented in Table I.

Analysing at the results, there is a noticable difference between backtracking algorithm and modal logic based algorithm. This is due to more time required to evaluate the formula at each iteration. This is clearly visible from the box plot in Fig. 7.

Having this in mind, both algortihms have similar algorithmic complexities, meaning that the complexity grows significantly with maze size. Backtracking runs faster, but it

TABLE I
AVERAGE TIME OF EXECUTION (TOE) IN USECONDS FOR EACH OF THE SCENARIOS

	Backtracking algorithm ToE (us)	Modal logical algorithm ToE (us)
11x11	9.95458	143.33611
21x21	14.50989	433.197
51x51	80.91908	3639.0306
101x101	116.90543	12072.385
201x201	463.33576	37226.919

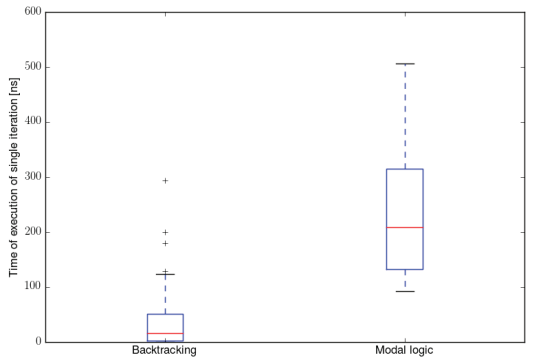

Fig. 7. Time of execution in nseconds of a single iteration for both backtracking algorithm and modal logic based algorithm

was optimised for only this class of problems. Modal logic with LogicToolbox, on the other hand, can be employed to model and solve wider class of decision making problems which is the main benefit of this kind of abstraction.

IV. CONCLUSION

In this paper we presented the modal logic based approach to the maze solving problem. First, we introduced a novel toolbox called LogicToolbox. This framework is freely available under open source MIT Licence. We used this framework to solve a typical decision making problem. For this paper, we chose the maze solving problem.

The problem was constructed as a satisfiability test for a modal logic formula. The formula is satisfiable in only one context, or set of worlds in which formula is valid. This gave out the solution to the maze problem which generates path from starting position to the predefined goal position. The main drawback was that the performance deteriorated, but this is usually an expected result when implementing abstract reasoning for decision making processes.

For future work, we plan on extending LogicToolbox with other types of modal logics and operators. This will further enhance the design of decision making processes that involve linguistic modalities such as belief, time, and knowledge.

REFERENCES

[1] N. S. Rao, S. Kareti, W. Shi, and S. S. Iyengar, "Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms," Oak Ridge National Lab., TN (United States), Tech. Rep., 1993.

[2] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference on*. IEEE, 2011, pp. 1067–1069.

[3] P. Modesti and A. Sciomachen, "A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks1," *European Journal of Operational Research*, vol. 111, no. 3, pp. 495–508, 1998.

[4] D. A. Crowe, B. B. Averbeck, M. V. Chafee, J. H. Anderson, and A. P. Georgopoulos, "Mental maze solving," *Journal of Cognitive Neuroscience*, vol. 12, no. 5, pp. 813–827, 2000.

[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[7] M. J. Fuerstman, P. Deschatelets, R. Kane, A. Schwartz, P. J. Kenis, J. M. Deutch, and G. M. Whitesides, "Solving mazes using microfluidic networks," *Langmuir*, vol. 19, no. 11, pp. 4714–4722, 2003.

[8] O. Steinbock, Á. Tóth, and K. Showalter, "Navigating complex labyrinths: optimal paths from chemical waves," *Science*, vol. 267, no. 5199, pp. 868–871, 1995.

[9] T. Nakagaki, H. Yamada, and Á. Tóth, "Intelligence: Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 6803, p. 470, 2000.

[10] D. Osmanković and S. Konjicija, "Implementation of q - learning algorithm for solving maze problem," in *MIPRO, 2011 Proceedings of the 34th International Convention*. IEEE, 2011, pp. 1619–1622.

[11] V. S. Gordon and Z. Matley, "Evolving sparse direction maps for maze pathfinding," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1. IEEE, 2004, pp. 835–838.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[13] S. Mishra and P. Bande, "Maze solving algorithms for micro mouse," in *Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on*. IEEE, 2008, pp. 86–93.

[14] E. Pacuit, *NEIGHBORHOOD SEMANTICS FOR MODAL LOGIC*. Springer, 2017.

[15] A. Saffidine and T. Cazenave, "A general multi-agent modal logic k framework for game tree search," 2012.

[16] W. van der Hoek and M. Pauly, "Modal logic for games and information," in *Studies in Logic and Practical Reasoning*. Elsevier, 2007, vol. 3, pp. 1077–1148.

[17] S. A. Kripke, "Semantical analysis of modal logic i normal modal propositional calculi," *Mathematical Logic Quarterly*, vol. 9, no. 5-6, pp. 67–96, 1963.

[18] O. Gasquet, A. Herzig, B. Said, and F. Schwarzentruber, *Kripke's Worlds - An Introduction to Modal Logics via Tableaux*, ser. Studies in Universal Logic. Birkhäuser, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-7643-8504-0

[19] E. Dijkstra, "Algol 60 translation : An algol 60 translator for the x1 and making a translator for algol 60," jan 1961.

[20] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[21] M. Fitting, "First-order modal tableaux," *Journal of Automated Reasoning*, vol. 4, no. 2, pp. 191–213, 1988.