university of groningen / faculty of science and engineering

Simulations of zero-one laws in modal logics GL, S4 and K4

Bachelor's Project Thesis

Jakub Zbigniew Łucki, s3986209, j.z.lucki@student.rug.nl,
Supervisors: Prof. Dr. Rineke Verbrugge

Abstract: A zero-one law is a property of modal logics stating that any formula is either almost always valid or almost always invalid in a given modal logic. This property can concern both model validity and frame validity. Furthermore, it has been shown that all modal logics obey zero-one laws concerning model validity. However, in the case of frame validity, only a modal logic **GL** has been successfully proven to have a zero-one law, while several others, such as modal logics **S4** and **K4**, are only hypothesised to obey it. This study aims to empirically confirm the proof for frames and models in **GL**, and check whether zero-one laws hold for frames and models of **S4** and **K4**. To accomplish this, each of 8047 distinct formulas was validated in 5000 randomly generated models and 500 randomly generated frames of each modal logic. Kleitman and Rothschild's result about the structure of almost all finite partial orders was used as a base for generating models and frames. The experiment's results suggest that the above-mentioned modal logics obey zero-one laws for both models and frames.

1 Introduction

Model theory is a field at the intersection of mathematics and philosophy, which is concerned with formal languages and their interpretations (Hodges, 2022). Syntactical rules of a language are nothing more than a framework. Therefore, to make any inferences about their semantics we need to infuse them with context or in different words interpret them. Such interpretation results in a model, which sets the context for particular situation. As a consequence we can deliberate about it through the notion of model-theoretic truth.

Models are not restricted in size by definition and many techniques in classical logic assume them to be infinite (Hodges, 2022). However, infinite structures are infeasible in real-world applications. Therefore, to make use of models the study of finite structures is needed (Immerman, 2012, p.6).

A finite structure contains a universe $\mathcal{A}$ which is a finite set, and a vocabulary σ which contains possible relations $P, Q, R, ...$ with associated arities and possibly constants $c, d, ...$ (Ebbinghaus & Flum, 2005). Examples of finite structures include graphs, orders and most importantly for us finite Kripke frames and finite Kripke models.

One of the most intriguing properties of such structures are zero-one laws. In probability theory, zero-one laws refer to events the occurrence of which is determined by random variables with probabilities of either 0 or 1 (Prokhorov, 2011). However, the zero-one laws which hold only for a subset of all finite structures are slightly different, since they are concerned with limits rather than raw probabilities.

The zero-one law for finite structures is defined as follows (adapted from Verbrugge, 2021). Let L be a vocabulary such as first-order logic without function symbols and let $A_n(L)$ be the set of all labelled L-structures with universe $1, ..., n$. Now let $\mu_n(\sigma)$ indicate the fraction of members of $A_n(L)$ in which formula σ is true, that is:

$$\mu_n(\sigma) = \frac{|M \in A_n(L) : M \models \sigma|}{|A_n(L)|},$$

where M is a model or in other words an L-structure. Then for every $\sigma \in L$, either $lim_{n\to\infty}\mu_n(\sigma) = 0$ or $lim_{n\to\infty}\mu_n(\sigma) = 1$. We can refer to this as a formula being almost surely valid or almost surely not valid with respect to models. However, we can also formulate this law for frame validity, by changing the meaning of $A_n(L)$ to the set of all possible frames, where L indicates

a modal language, and letting M indicate a particular frame.

The zero-one law has been proved for finite structures in first-order logic without function symbols first by Glebskii, Kogan, Liogon'kii, and Talanov (1972), and then independently, by Fagin (1976). Since that time, zero-one laws have been examined for other, more expressive, logics such as monadic existential second-order logic (Kaufmann & Kaufmann, 1988), where it does not hold.

Halpern and Kapron (1994) have shown that a zero-one law for Kripke models follows from its counterpart for first-order logic via the translation method introduced by Van Benthem (1984). However, this translation does not allow to conclude zero-one laws for frame validity, nor does it provide axiomatizations for model validity.

For these purposes, separate attempts have been made. Halpern and Kapron (1994) proved that zero-one laws hold for model validity in modal logics **K**, **T**, **S4**, **S5**. Their paper also covers the proofs for zero-one laws regarding frame validity, together with corresponding axiomatizations. However, the axiomatization and proof for **K** was rendered false by Le Bars (2002), who has shown that no zero-one law holds for frames in modal logic **K**. Furthermore, the proof and axiomatization for **S4** has been found invalid by Verbrugge (2021). Ultimately, the proof and axiomatization for **T**, could also be prone to mistakes given previous errors, but the case is unsettled (Verbrugge, 2021).

Given the proof for zero-one laws for first-order logic restricted to partial orders (Compton, 1988) and the possibility of translating a modal language to it (Van Benthem, 1984), the zero-one law for model validity in provability logic (**GL**) follows. Nevertheless, Verbrugge (2021) provided a stronger proof for this property as well as proved it with respect to frame validity. Moreover, the author has axiomatized the sets of almost surely valid formulas with respect to both models and frames.

Verbrugge (2021) used the combinatorial result by Kleitman and Rothschild (1975), which provides a very specific structure of reflexive partial orders, and extended it to irreflexive ones. Kleitman and Rothschild's result was previously used in a proof by Compton (1988). Moreover, Halpern and Kapron (1994) have proven that almost surely every transitive and reflexive relation is a (non-strict) partial order, thus making the combinatorial result applicable to a wider range of modal logics.

Modal logic itself has applications in, among others, linguistics, economics and computer science (Grädel et al., 2007). Furthermore, the abovementioned zero-one laws and other limit laws have applications in database theory (e.g. Halpern, 2006). General asymptotic probabilities, their relation to default reasoning and degrees of belief have been of great interest in the field of AI (Verbrugge, 2021).

So far there have been only theoretical proofs for zero-one laws in modal logics. However, an empirical approach, while not providing a strong proof, has several advantages. First it can verify existing proofs. Secondly, it can provide insights and inspiration for future proofs and axiom systems. For instance, proofs for modal logics **S4** and **K4** are being developed currently (Verbrugge, 2021).

Therefore, this paper will focus on the following research question: "Do simulations of formulas in models and frames confirm zero-one laws for **GL**, **S4** and **K4** modal logics?" The last two languages were chosen to aid the proof development, whereas **GL** was selected to verify the proof of (Verbrugge, 2021).

To answer this question, a random formula generation was built, then the generated formulas were checked in models and frames using a model checker. Modal checking is a vast field on its own, concerning verification of systems with respect to their specifications (Clarke & Schlingloff, 2001). For this purpose many model checkers have been built (for comprehensive list see: Wikipedia contributors, 2022). However, they primarily use very purpose-specific languages such as Computational Tree Logic or Temporal logic. As a consequence, there is lack of model checkers for standard modal logics such as **K** (Clarke & Schlingloff, 2001). Therefore, a dedicated model checker had to be developed.

1.1 Partial orders

A binary relation R (for instance $\leq$) is called a "partial order" on a set S if it is:

- *reflexive*: xRx,
- *antisymmetric*: if xRy and yRx, then $x = y$,
- *transitive*: if xRy and yRz, then xRz,

for each $x, y, z \in S$.

There is also a variation of partial orders indicated by the preceding adjective "strict". Relations of this type (for instance $<$) are:

- *irreflexive*: not xRx,
- *asymmetric*: if xRy, then not yRx,
- *transitive*: if xRy and yRz, then xRz,

for each $x, y, z \in S$ (see for instance Wallis, 2013).

1.2 Kleitman and Rothschild's combinatorial result

Kleitman and Rothschild (1975) have shown that with asymptotic probability 1 all finite partial orders have a very specific structure. This result holds for both strict and non-strict partial orders. Firstly, there are no chains $aRbRcRd$ of more than 3 elements.

Then, let us denote the number of elements in a partial order with n. These elements can be partitioned into 3 layers:

L_1 - set of minimal elements of asymptotic size $\frac{n}{4}$,

L_2 - set of elements immediately succeeding elements in L_1 of asymptotic size $\frac{n}{2}$,

L_3 - set of elements immediately succeeding elements in L_2 of asymptotic size $\frac{n}{4}$.

Each element in L_1 has 50% chance to be related to a given element in L_2, therefore, every element in the minimal set is connected to asymptotically half of the elements in L_2. Similarly, each element in L_2 has 50% chance to be related to a given element in L_3.

1.3 Modal logic

Let us inductively define modal logic $L(\Phi)$ over the finite set of propositional atoms $\Phi = \{p_1, ..., p_k\}$ for some natural number k:

1. If $p \in \Phi$ then $p \in L(\Phi)$.
2. If $A \in L(\Phi)$ and $B \in L(\Phi)$, then also $\neg A \in L(\Phi)$, $\Box A \in L(\Phi)$, $\Diamond A \in L(\Phi)$, $(A \wedge B) \in L(\Phi)$, $(A \vee B) \in L(\Phi)$, $(A \rightarrow B) \in L(\Phi)$ and $(A \leftrightarrow B) \in L(\Phi)$,
3. Nothing is a formula of $L(\Phi)$ unless it was created through finitely many repeated applications of 1 and 2.

Moreover, a Kripke frame is a pair $F = (W, R)$, where $W = \{1, 2, ..., n\}$ is a non-empty set of worlds and $R \subseteq W \times W$, is a binary accessibility relation. A Kripke model $M = (W, R, V)$ contains a frame $F = (W, R)$ and a valuation V, which assigns a truth value 1 or 0 to each atomic proposition in each world. The truth definition is as usual in modal logic, including the clause:

$$M, w \models \Box\varphi \text{ if and only if for all } w' \text{ such that } wRw', M, w' \models \varphi.$$

A formula φ is valid in model M if and only if for all $w \in W$, $M, w \models \varphi$.

A formula φ is valid in frame F if and only if for all valuations V and for all worlds $w \in W$, $F, V, w \models \varphi$ (definitions adapted from Verbrugge, 2021).

1.4 Types of modal logic

All types of modal logic used in this paper extend the **K** system, which has the following axioms (Garson, 2021):

All (instances of) propositional tautologies
$\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$

Moreover, the rules of inference include modus ponens and necessitation:

if $\mathbf{K} \vdash \varphi \rightarrow \psi$ and $\mathbf{K} \vdash \varphi$, then $\mathbf{K} \vdash \psi$,
if $\mathbf{K} \vdash \varphi$, then $\mathbf{K} \vdash \Box\varphi$.

This system does not impose any restrictions on its frames.

1.4.1 Provability logic

This modal logic (**GL**) extends **K** though the following axiom (Verbrugge, 2017):

$$\Box(\Box\varphi \rightarrow \varphi) \rightarrow \Box\varphi$$

Moreover, provability logic is sound and complete with respect to all transitive and converse well-founded (there is no infinitely ascending sequence

$x_1Rx_2Rx_3...$) frames. From these two constraints a third follows: irreflexivity.

Given these constraints, we can conclude that sufficiently large finite frames of **GL** are strict partial orders and therefore, will have the structure described by Kleitman and Rothschild (1975).

1.4.2 System K4

This system imposes the transitivity constraint on its frames (Garson, 2021). Therefore, it extends **K** by the following axiom:

$$\Box\varphi \rightarrow \Box\Box\varphi$$

1.4.3 System S4

This system imposes reflexivity and transitivity constraints on its frames (Garson, 2021). Therefore, it extends the **K** system with the following axioms, respectively:

$$\Box\varphi \rightarrow \varphi$$
$$\Box\varphi \rightarrow \Box\Box\varphi$$

Given that Halpern and Kapron (1994) have proven that almost surely every transitive and reflexive relation is a partial order, we can represent sufficiently large finite frames of this system using the result by Kleitman and Rothschild (1975).

2 Problem analysis

In modal systems "validity in all finite models" coincides with "validity in all finite frames", however, this is not the case for "almost sure validity" as there are many more formulas that are almost surely valid in models than in frames (Verbrugge, 2021). Therefore, this work will focus on six subproblems: almost sure validity in models and frames for **GL**, **K4**, and **S4**. Given different characteristics of each problem, the Kleitman and Rothschild's (henceforth, KR) result needs to be adjusted accordingly.

Across all subproblems, elements in the KR result correspond to worlds in modal logic and a relation corresponds to the accessibility relation. However, transitivity cannot be directly translated from the KR result to modal frames. In the first case it is implicit, while in the latter case we have to make it explicit by introducing an accessibility relation from the worlds in layer L_1 to worlds in L_3, which are reachable via layer L_2.

Apart from being transitive, provability logic is also converse well-founded. The combination of these constraints entails irreflexivity. Therefore, to match these constraints irreflexive Kleitman and Rothschild structures have to be used for simulating formulas in **GL**.

In case of system **S4**, we need to additionally impose the reflexivity constraint on the partial order. This can be done by introducing a reflexive accessibility relation in each world.

System **K4** is only transitive. To accommodate undetermined reflexivity we can introduce reflexive relation in exactly half of the worlds in each layer, as suggested by Verbrugge (n.d.).

2.1 Exhaustive approach

Thanks to KR result we do not have to generate all possible Kripke models and Kripke frames to check almost sure validity. Instead, we can generate models and frames based on the KR structure. However, even then the cardinality of the set of all structures ($|A_n(L)|$) is enormous.

A model in a modal language consists of a set of worlds (of cardinality n, given during generation), a relation and a valuation function. Usually, a relation refers to a set of ordered pairs indicating which worlds can access which, however, for simplicity of the explanation, in the reminder of this chapter I will refer to each ordered pair as a relation. Therefore, the total number of possible models will be the product of the number of all possible combinations of relations and the number of all possible valuation functions.

A combination of a structure's relations is a unique subset of all possible relations in a structure. The total number of relations in a KR structure consisting of n worlds is

$$2 \times 2m \times m = 4m^2,$$

where m ($= \frac{n}{4}$) is the number of worlds in layers L_1 and L_3 each, and $2m$ ($= \frac{n}{2}$) is the number of worlds in layer L_2.

As a consequence, the number of possible combinations of relations in this case is 2^{4m^2}, because each relation has two possible states: present and

absent. It is important to mention that the accessibility relations resulting from the transitivity constraint do not affect these numbers because they are fully dependent on the arrangement of other relations in the structure.

Let $\Phi = \{p_1, ..., p_k\}$ be a set of propositional atoms with cardinality k. Then the number of possible valuations will be equal to $(2^k)^{4m}$, where $m\ (= \frac{n}{4})$.

Therefore, the number of all models of a given size is

$$2^{4m^2} \times (2^k)^{4m} = 2^{4m^2+4km}.$$

This is a number of models in which one would have to check a formula's validity in order to determine whether it is almost surely valid*.

A frame in a modal language consists only of a set of worlds (of cardinality n, given during generation) and a set of relations. Therefore the number of frames in which a formula's validity needs to be checked in order to determine almost sure validity is 2^{4m^2}. Although, there are less frames to generate in total than models, to check a formula's validity in a given frame we need to check all possible valuations on this frame, which effectively multiplies the previous number by $(2^k)^{4m}$.

2.2 Infinite approach

In the previous section we considered generating all possible structures, which turned out to be impractical due to exponential growth. We can partially alleviate this problem by precluding structures containing features which will disappear as the number of worlds grows larger. The features were extracted from the work of (Verbrugge, 2021). They are as follows:

- *Dead-ends* - This feature describes worlds in layers L_1 and L_2 from which there are no outgoing accessibility relations to higher layers. It can be easily seen that as the number of worlds increases the fraction of structures containing *dead-ends* decreases.

- *Dead-starts* - This feature describes worlds in layers L_2 and L_3 to which there are no incoming accessibility relations. Again, as the number of worlds increases the fraction of structures containing *dead-starts* decreases.

*In fact to check it one would have to repeat the procedure for several sizes of models, or choose a very large n.

By using these features we filter out improbable structures. As a consequence we enforce that every world in L_1 can reach at least one world in layer L_3, which was also given in (Verbrugge, 2021).

Now we can reassess the number of combinations of relations in a structure of size n. The approach to compute that number is to subtract number of relation combinations containing one of the abovementioned features and subtract that number from the total number of possible relation combinations.

First, lets focus on the relations between layers L_1 and L_2. *Dead-ends*, in the L_1 layer, can occur in a single world, but also in all of them. To capture all arrangements in which *dead-ends* can appear we can use the following summation:

$$\sum_{i=0}^{m} \binom{m}{i}.$$

To capture the number of arrangements in which *dead-starts* occur in combination with all arrangements of *dead-ends* we can use this summation:

$$\sum_{i=0}^{m} \sum_{j=0}^{2m} \binom{m}{i} \binom{2m}{j}.$$

Now we have a way to compute all arrangements of *dead-ends* together with *dead-starts*. However, this formula does not cover all the possible arrangements of the relations which are connecting worlds which are neither *dead-ends* nor *dead-starts*. To account for all such arrangements we first need to define how many relations there can be in between the first two layers given a number i of *dead-ends* and a number j of *dead-starts*. There is maximum of $(m-i)(2m-j)$ of such relations. Now we need to define how many relations can be missing from the maximum without a risk of encountering an arrangement where there is an unaccounted *dead-ends* or *dead-starts*. We can capture this in a function $f(i,j) = \max(\min(m-i, 2m-j) - 1, 0)$. For example, if we have $m = 4,\ i = 1,\ j = 0$, so there is only one *dead-end* and no *dead-starts*, we would have $f(1,0) = 2$, hence we can have at most two relations missing from their maximum number.

To account for all the arrangements of *dead-ends*, *dead-starts* together with the relations between other worlds, we can define the function

$g(m)$:

$$\sum_{i=0}^{m}\sum_{j=0}^{2m}\binom{m}{i}\binom{2m}{j}\left(\sum_{k=0}^{f(i,j)}\binom{(m-i)(2m-j)}{(m-i)(2m-j)-k}\right)$$

However, $g(m)$ also accounts for the arrangement where there is no *dead-ends*, nor *dead-starts*, namely when $i = 0,\ j = 0$. To offset this we need to subtract the following from $g(m)$:

$$\sum_{l=0}^{m-1}\binom{(m)(2m)}{(m)(2m)-l}.$$

Finally, we can subtract this result from the number of all possible combinations:

$$2^{m\times 2m} - \left(g(m) - \sum_{l=0}^{m-1}\binom{(m)(2m)}{(m)(2m)-l}\right).$$

However, this only takes into account relations between layers L_1 and L_2 and we also need to consider the ones between layers L_2 and L_3. To do that, we need to realise that if we rotate a structure by 180° and look at relations between layers L_2 and L_3, *dead-ends* become *dead-starts* and vice versa. Therefore, they are symmetrical. As a consequence, the result above also gives the number of combinations of the relations between layers L_2 and L_3. Hence, we can simply square the result above as a way to compute all the combinations of the two sets of relations:

$$\left(2^{m\times 2m} - \left(g(m) - \sum_{l=0}^{m-1}\binom{(m)(2m)}{(m)(2m)-l}\right)\right)^2.$$

3 Implementational details

This section covers the design of the algorithms and data structures used throughout the experiments. Every functionality will be explained in a dedicated module. We treat them as building blocks and, thus, they will be presented in a bottom-up way where every consequent module depends on the previous ones.

This project has been implemented in the `Julia` programming language version `1.7.2`. It is a relatively new language, which tends to be faster than more common alternatives such as `Python` (see for example, Lin, 2020; Moura, 2021). Given that the main experiment is computationally expensive, speed was one of the most significant objectives when selecting a language. Moreover, `Julia` has a significant support and contains all features of a modern language, which is not necessarily the case for the `C` programming language. The code used for this project can be accessed on `Github` using the following link: `https://github.com/J4Q8/zero-one-laws`.

During development of the experiment's infrastructure, I noticed that `Julia` does not have any advanced facilities that could be directly used for logic research. Therefore, the code used for this project has been adapted for more general use and transformed to a package `LogicToolkit` in a hope to facilitate future logical simulations. It can be installed from general registry of `Julia` packages. However, it can also be accessed directly through the following link: `https://github.com/J4Q8/LogicToolkit.jl`.

3.1 Binary Trees

Propositional and modal operators have arity of at most 2. Moreover, operators have to have a specified hierarchy, in order to avoid ambiguity. Therefore, the choice of binary trees, which provide the two abovementioned features, as a representation of formulas is a natural choice. For instance, the main axiom of **GL**, $\Box(\Box\varphi \to \varphi) \to \Box\varphi$ would be represented as shown in Appendix A.

Another design choice was to always make unary connectives right children of a binary node. Due to this decision, inorder traversal returns a formula in a string format.

However, using this data structure has one downside. It cannot explicitly express associativity and commutativity. To overcome this, special functions had to be designed, which for example return all conjuncts of a given formula.

3.2 Parser

To make the user experience easier, a parser was built. First it replaces connective symbols which are easy to type with their correct versions. Therefore, for instance "^" becomes "∧" and "v" becomes "∨". The complete list of accepted symbols is in

Appendix B. Input transformed in this way is then parsed using the grammar shown in Appendix C. If a formula follows this grammar, it will be converted into a binary tree, through a series of recursive calls.

3.3 Simplifier

The functionality of this module entails the ability to reduce formulas to their least trivial form. To perform this reduction, a set of formulas containing truth values and their equivalences was created. Using them, we can remove trivialities in formulas. The subset of commutative rules, for which the location of a truth value ($\bot, \top$) in the original formula does not matter, is the following:

$$\begin{aligned} \varphi \wedge \top &\Leftrightarrow \varphi & \varphi \wedge \bot &\Leftrightarrow \bot \\ \varphi \vee \top &\Leftrightarrow \top & \varphi \vee \bot &\Leftrightarrow \varphi \\ \varphi \leftrightarrow \top &\Leftrightarrow \varphi & \varphi \leftrightarrow \bot &\Leftrightarrow \neg\, \varphi \end{aligned}$$

Then, the subset of equivalences for which the order of the atoms is fixed is as follows:

$$\begin{aligned} \varphi \rightarrow \top &\Leftrightarrow \top & \varphi \rightarrow \bot &\Leftrightarrow \neg\, \varphi \\ \top \rightarrow \varphi &\Leftrightarrow \varphi & \bot \rightarrow \varphi &\Leftrightarrow \top \\ \neg\bot &\Leftrightarrow \top & \neg\top &\Leftrightarrow \bot \\ \Box\top &\Leftrightarrow \top & \Diamond\bot &\Leftrightarrow \bot \\ \neg\neg\varphi &\Leftrightarrow \varphi \end{aligned}$$

However, it is also possible for trivialities to arise without truth values only due to the interplay of formulas. To remove them, the following set of equivalences was formed, where the original formulas are commutative:

$$\begin{aligned} \varphi \wedge \varphi &\Leftrightarrow \varphi & \varphi \wedge \neg\, \varphi &\Leftrightarrow \bot \\ \varphi \vee \varphi &\Leftrightarrow \varphi & \varphi \vee \neg\, \varphi &\Leftrightarrow \top \\ \varphi \leftrightarrow \varphi &\Leftrightarrow \top & \varphi \leftrightarrow \neg\, \varphi &\Leftrightarrow \bot \end{aligned}$$

These sets of equivalences were selected, because they provide an optimal balance between application complexity and a level of captured triviality. One could further enhance the simplifier with distributive law, De Morgan's laws, or *modus ponens* and *modus tolens*. However, this would significantly harm the performance of this algorithm which needs to be frequently applied throughout the experiment desribed in Section 4. Moreover, these enhancements can be seen as logical inference rather than removing triviality from formulas. Hence, they were not implemented in this module.

The next challenge in developing the simplifier was combination of associativity and commutativity, which is the case for the conjunction, disjunction and biconditional operators. Given that binary trees are not able to express these properties innately, it could be the case that for instance $(\varphi \wedge \psi) \wedge \neg\, \varphi$ would not be reduced to $\bot$ because φ and $\neg\, \varphi$ are not children of the same node. Therefore, whenever one of the abovementioned operators is encountered all of its sub-elements, let us call them juncts, connected through repetitions of a given operator will be found. For example, the formula $(\varphi \wedge \psi) \wedge \neg\, \varphi$ would result in three juncts: $\varphi, \psi, \neg\, \varphi$.

The simplifier oftentimes needs to check whether two formulas are equivalent. This is done by comparing the set of juncts of one formula with the set of juncts of the other formula. If the two sets are equal, meaning that all juncts in both sets have an equivalent junct in the other set, then the two formulas are equivalent. This method is robust with regards to associativity and commutativity, which means that, for instance, formulas $(\varphi \vee \sigma) \vee \psi$ and $(\psi \vee \varphi) \vee \sigma$ will be recognized as equivalent.

During the simplification process, every subtree of a formula is broken down into juncts (formula can have only one junct, e.g. $\Box\varphi$). Then all juncts are replaced with equivalent simplified formulas. This step is repeated until no new replacement can be made. As a consequence, resulting formula contains no repeating juncts and contains no trivialities as given in lists of equivalences.

To improve the performance of the algorithm, the equivalences that result in a truth value are tried first. For example, when disjunction is the main connective of a formula, we first check if any of its juncts is $\top$ or if there are two opposite formulas such as φ and $\neg\, \varphi$. This allows to avoid redundant recursive calls.

3.4 Formula generator

The computational cost per formula of the experiment described in Section 4 is high. Therefore, it was imperative to test original and potentially interesting formulas, the truth valuation of which is not trivial, which could be the case for completely

stochastic formula generator. To avoid it, the developed formula generator relies heavily on the simplifier and underlining principles introduced in Section 3.3.

Additionally, this generator is able to produce formulas of arbitrary depth and arbitrary maximum modal depth. The former indicates the depth of the binary tree of a formula, with the root node counting as 1. The latter indicates the deepest nesting of modal operators (MD) in a formula and is defined inductively as follows:

1. $MD(p_i) = 0$, for all propositional atoms p_i;
2. $MD(\varphi \oplus \psi) = maximum(\{MD(\varphi), MD(\psi)\})$, where $\oplus \in \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$;
3. $MD(\otimes\varphi) = MD(\varphi) + 1$, where $\otimes \in \{\Box, \Diamond\}$.

The formulas are generated in a top-down manner. Thus, the first operator to be selected will be a main connective of a new formula. Then, the children of the main connective are generated in a recursive manner. At the end of each step, the resulting intermediate formula is reduced to its simplest form via the simplifier. This ensures originality of each formula.

A secondary objective for this formula generator was to produce a reliable sample (free from bias) of all interesting formulas. Therefore, at every step the operator was chosen from the set of available symbols using uniform distribution. The original set contains all operators, all atoms and truth values. However, the set of available symbols changes dependently on the directly higher connective, maximum modal depth and maximum formula depth. To improve performance of this algorithm and make the simplifier's job easier, certain operators are banned given their predecessors. Therefore, negation cannot have another negation as a parent and truth values cannot appear in trivial scenarios specified in Section 3.3. Once a required formula depth is reached, the algorithm is forced to place a random atom and thus stop building the tree there. Moreover, modal operators are a subset of the available symbols set until the maximum modal depth is reached.

It is important to notice that given the uniform distribution, it is possible that a formula tree will be closed (have only atoms in its leaves) before reaching the desired formula depth. In that case, the generator will be rerun until the desired depth is reached. Similar situation can happen with modal depth, but then the algorithm is not rerun, given that user can select arbitrary maximum model depth and not any particular model depth of a formula.

Once a formula is generated, it is checked against all previous formulas for equivalences. If there are none, it is added to the final list.

3.5 Tableau solver

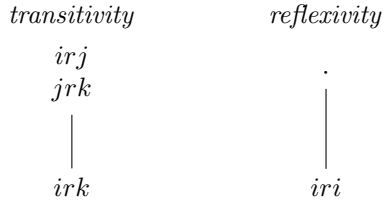
Throughout the project, a tableau solver for **GL**, **S4** and **K4** was built. This algorithm facilitated checking whether a given formula is a contradiction or a tautology in these three modal languages.

3.5.1 Semantic tableaux

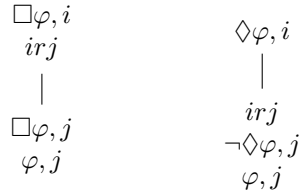
Semantic tableaux are a method used to check validity of an inference. Priest (2008, p. 56-60) has shown that the method of semantic tableaux is sound and complete with respect to the semantics of modal language $\mathbf{K}_\rho$, which has the reflexivity constraint, and modal language **K4**, which has the transitivity constraint. The proof for soundness and completeness of this method with respect to the semantics of **S4** can be derived from the proofs for **K4** and $\mathbf{K}_\rho$. Semantic tableaux are also sound and complete with respect to provability logic (see for example Van Loo, 2017).

A tableau starts as an initial list containing all premises and the negation of the consequent of the inference, accompanied by a label for a world (usually 0). Then, by repeated application of rules we try to discover a counter-model. If none can be found, such tableau is said to be closed and the inference holds.

Since all three logics **GL**, **S4** and **K4** extend modal logic **K**, all rules for **K**-tableaux are applicable to them as well. A comprehensive list of tableau rules is given by Priest (2008). Moreover, **K4** is extended by a tableau rule for transitivity, whereas **S4** is extended by both transitive and reflexive rules given below:



where i, j, k are worlds introduced on the branch. To accommodate the converse well-foundedness constraint of **GL**, a special $\Box$-rule and $\Diamond$-rule were developed by Van Loo (2017). They are as follows:



In case of $\Box$-rule, j was already on the branch; however, for $\Diamond$-rule, a new j is introduced. These rules already incorporate the transitive constraint in **GL**, therefore, the extra transitivity rule does not have to be used for this logic.

It is important to note that there are also special rules available for the negations of these modal operators (for all three modal languages). They are not given here, because they are not used in the algorithm. Instead, to all such formulas $\Box\varphi \Leftrightarrow \neg\Diamond\neg\varphi$ equivalence is applied. This decision aims to simplify the implementation of $\neg$-rules.

One can use semantic tableaux to check whether a formula φ is a tautology by starting a tableau with initial list: $\neg\ \varphi, 0$. If the tableaux closes it means that φ is a tautology of the system.

To check whether a formula φ is a contradiction we can start a tableau with initial list $\varphi, 0$ If the tableaux closes in this case, it means that by using formula φ we can prove anything which is a clear indication of contradiction.

3.5.2 Design

There are multiple heuristics available, which are supposed to make the proof-search more efficient. One of them is converting input formulas into Negated Normal Form (NNF), which allows only conjunctions, disjunctions and negations of atoms. However, the advantage it was supposed to provide was found to be doubtful for formulas of various sizes (Van Loo, 2017).

Another heuristic is to convert the formulas to predicate logic. Then, for example $\Box\varphi$ becomes $\forall x(\neg wRx \vee \varphi_x)$. However, this introduces additional complexity which deviates the tableaux from its textbook form (Schwarz, 2022). To improve readability and transparency of this algorithm, I decided to use the form of textbook tableaux, so I did not use this improvement.

The tableau is solved in a depth-first search manner, so that one branch had to be closed to explore the other. This approach has been successfully applied by Van Gelder (2021) in his Bachelor thesis, which resulted in good performance. More importantly it avoided out-of-memory errors, which were prevalent in the breadth-first search approach used by Van Loo (2017).

The formulas on the initial list and the current branch were stored in an ordered list and for each formula, information is saved whether a rule has been applied to it or not. A second list stores all relations that have been introduced in the tableau along with the locations of the formulas which introduced them. A separate list stores formulas to be applied on a next branch, together with the position in the tableau where the next branch should be opened. Whenever a branch is closed, all formulas, relations that were introduced after the branching "fork" are removed and replaced with formulas that are supposed to be on the new branch.

The main list is then iterated, rules are applied to formulas which were not used so far and resulting formulas are added at the end of the list.This approach was chosen over sorting a main list for several reasons. First is that sorting formulas is time consuming, especially as their number increases and iterating the complete list cannot be completely avoided, since the $\Box$-rule needs to be checked every time a new relation is introduced. Secondly, it interferes with the clear chronology of applied rules, which can be problematic when switching to a new branch. Every iteration list is checked for contradiction, truth negation or two opposite formulas in order to close a branch as soon as possible.

To improve the performance and reduce the number of branches, all possible non-branching rules are applied first to the current branch and resulting formulas. This will stop only when there

are no more such rules to be applied. This is the case of all three modal languages. However, for **GL** all possible modal rules are applied before all branching rules. This decision was made due to converse well-foundedness of the provability logic, which precludes infinitely increasing chains of relations. Hence, modal rules can be applied first without a danger of encountering an infinite branch. For **S4** and **K4**, this is not the case, since they both are transitive, but not converse well-founded. Therefore, infinite branches are possible and not as infrequent as one would wish. Thus, in these two languages, modal rules and branching rules are applied in an alternating manner in order to balance the risk of encountering an infinite branch with the computational cost of having more branches to check.

Tableaux of some formulas are very large and complex to an extent where it is hard to determine whether a particular branch is actually infinite or just complex. Therefore, despite the infinite branch detector described in the next subsection a 30 second, time limit per inference was implemented. If a formula takes more than 30 seconds, then the algorithm treats such inference as inconclusive and the results are discarded.

3.5.3 Infinite branches

In order to make the tableau solver more versatile and robust, an infinite branch detector was implemented. However, infinite branches come in more shapes and sizes than expected. Therefore, the development of this detector took several iterations.

First, a naive assumption was made that on an infinite branch, formulas will be repeated in every new world. Thus, a simple pattern detector was built, which checked whether the set of formulas in the last world is a subset of the formulas in the previous world. Then the number of worlds for which this subset relation held was counted and tested against a threshold. Unfortunately, on the infinite branches, formulas do not have to repeat in every new world.

Therefore, a second assumption was made that on an infinite branch, formulas will repeat in new worlds periodically. This resulted in a second version of a pattern detector which allowed for formulas to be repeated periodically in new worlds. Although this improved the versatility of the infinite branch detector, on some branches formulas were indeed repeating but each formula had a different period.

The next approach was to completely abandon the apparent structure of a tableau and instead focus on how the worlds are related within a tableau. Therefore, relations from a given branch were analysed and orders of worlds were obtained. For instance, if there is $1r2, 2r3$ on a branch then the chain of worlds would be $\langle 1, 2, 3\rangle$. Then the previous two assumptions were tested not on a whole tableau, but on each chain individually. Unfortunately, this approach failed to capture the nature of the infinite branches.

The idea of chains and the first two assumptions were also tried together with the idea that infinite branches are directly caused by repeating $\Diamond$ formulas, since they are the ones opening new worlds. Nevertheless, some formulas still had infinite branches which were not captured by this algorithm. Moreover, at this point the checking whether a branch is infinite was significantly slowing down the algorithm.

Therefore, a last algorithm was tried which counts the number of newly introduced worlds which do not include at least one unique formula. By unique, I mean not appearing previously on a tableau (possibly in a different world). If there is no such unique formula in the last introduced 100 worlds, then such a branch is deemed infinite. This relatively light-weight solution provided at least as good results as the complex pattern checkers discussed above.

3.6 Model/frame generator

KR structures are represented as directed graphs using `Julia` package `LightGraphs`. The valuation function was stored in an array of dictionaries where each element corresponded to a single world.

The relations between the bottom layer (with m worlds) and middle layer (with $2m$ worlds) can be represented as Boolean adjacency matrix $\boldsymbol{M_1} \in \mathbb{R}^{m\times 2m}$. Similarly, relations between middle layer (with $2m$ worlds) and upper layer (with m worlds) can be captured in a Boolean adjacency matrix $\boldsymbol{M_2} \in \mathbb{R}^{m\times 2m}$. In both $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$, rows indicate a world with outgoing relations and columns indicate worlds with incoming relations.

Adjacency matrices facilitate unbiased randomness when generating structures, because they can be easily sampled from a uniform distribution. Once $\boldsymbol{M_1}, \boldsymbol{M_2}$ are sampled, they are checked whether they do not contain any dead-ends or dead-starts (nodes in middle and higher layer that do not have any incoming relations). This is done by verifying that there is no column, nor row, which contains only 0s.

All three modal languages require transitive closure. It could be naively computed through depth-first search, but this method becomes inefficient as matrices get larger. Therefore, a new method was developed, which allows the transitive closure adjacency matrix $\boldsymbol{M_3}$ to be computed via:

$$\sigma(\boldsymbol{M_1}\boldsymbol{M_2}),$$

where σ is a threshold function:

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & otherwise \end{cases}$$

Given the way matrix multiplication works, every entry in $\boldsymbol{M_3}$ is the thresholded result of a dot product of a row of $\boldsymbol{M_1}$ and a column of $\boldsymbol{M_2}$. This dot product represents all possible ways a given world in the upper layer is reachable from a given world in the lower layer. Therefore, if this dot product is not zero, it means that there is at least one path to reach that world.

Additionally, for frames in **S4**, the reflexive connections are added to all worlds and in **K4** the reflexive connections are added to exactly half of worlds in each layer. This ends the process of frame generation.

To generate a model given a frame, each world gets a random valuation function sampled from a uniform distribution.

3.7 Model checker

The purpose of the model checking algorithm is to check whether a formula is valid in a given model. In order to check validity, a formula has to hold in every world of the model. This is done via exhaustive search in order to ensure the lack of false positives. As a consequence, this algorithm becomes a bottleneck of the experiment described in Section 4.

To maximize its performance, several heuristics were used.

- *Early stopping* - once a formula was found invalid in a single world, we can conclude that it is invalid in a whole model.

- *Caching formulas* - once an intermediate formula is evaluated in a given world, its truth value in that world is stored. Given network-like structure of KR frames, formulas are often evaluated multiple times in the same world. Therefore, instead of evaluating it multiple times, the following evaluations will simply use cached truth values. This heuristic has been inspired by model checker introduced in Lagniez, Le Berre, de Lima, and Montmirail (2016).

- *Upper layer first* - formulas are checked in worlds of the upper layer first. This heuristic is supposed to quickly invalidate formulas of structure $\Diamond\varphi$ in frames of **GL** and **K4**. Since in these languages there are worlds in the upper layer which do not have any outgoing relations, formulas with $\Diamond$ as a main connective can be deemed invalid almost immediately.

3.8 Frame checker

In order to exhaustively check whether a formula is valid in a given frame, one would have to generate all possible valuations and evaluate them using the model checker. However, I have shown in Section 2 that this is not feasible as the number of valuations increases exponentially with the number of atomic propositions and number of worlds in a frame.

To overcome this limitation, I propose a probabilistic algorithm: x random valuations on a frame will be generated, then every one of them will be evaluated. Given a valuation and a frame, there can be only two outcomes of such evaluation: a formula can be either valid or invalid. If a formula is found invalid in one of the valuations on a frame, then we can conclude that it is invalid on a frame.

To determine the error probability of this method, we need to realise that on average, a given formula is valid in less than $\frac{1}{2}$ of all models. For each formula φ, at most one of φ and $\neg\varphi$ is valid in a model. Hence, $\leq \frac{1}{2}$ of all formulas are valid in a model. Furthermore, there are formulas φ for which

neither φ, nor $\neg\varphi$ are valid in a model. For example, the formula φ can be one of the propositional atoms, then for φ or $\neg\varphi$ to be valid in a model, all worlds would have to have the same valuations for that propositional atom, which is highly unlikely for large models. Therefore, strictly less than $\frac{1}{2}$ of all formulas are valid in a model. Then if we pick a formula and a random valuation on a frame, the chance that the formula is valid in that model is less than $\frac{1}{2}$, on average. Therefore, a chance that an invalid formula is found valid on a frame, because of the unfortunate sample of random valuations is less than $(\frac{1}{2})^x$.

3.9 Tests

All described modules have been profoundly tested either directly or indirectly.

The tableau solver has been tested using verified inferences compiled from Priest (2008), Van Gelder (2021) and self-verified formulas generated by the formula generator (51 inferences in total). The examples include inferences in all three languages, closed and open tableaux, as well as formulas which result in infinite branches.

The simplifier was tested on 41 cases, which were carefully crafted to represent multiple trivialities of varying extent.

Model and frame checkers were tested on 62 formulas each, which included tautologies and contradictions of the three modal logics. Some of the formulas overlap with the test cases for the tableau solver, whereas the others were randomly generated and checked to be contradictions or tautologies.

All inferences and formulas used as test cases can be found on the project's website on `Github`.

4 Experimental Design

For the experiment, 8000 formulas of depth 6 to 13 were generated based on the set of two propositional atoms: $\{p, q\}$. These parameters were selected in order to generate interesting and complex formulas, but short enough to be comprehensible. All of the formulas had maximal modal depth of five. Given that KR structures have only 3 layers, greater depth did not seem to provide any benefits.

If a formula was a contradiction or a tautology in all three modal languages, then it was excluded from the experiment. Tautologies and contradictions inherently support zero-one laws, thus it is redundant to check whether they are almost surely valid. If a formula was not a tautology or a contradiction in at least one of the languages, this information would lay foundations for several of the analysis made in the Section 5.

Additionally, 47 formulas were selected based on their special meaning. For instance, axioms for almost sure model validity introduced by Verbrugge (2021) are on this list. The complete set of these formulas can be found in the Appendix D.

Zero-one laws are based on a limit therefore, trends are more important than absolute values. In order to check these trends, each experiment will be performed on KR structures consisting of 40, 48, 56, 64, 72, 80 worlds. Henson, Rideout, Sorkin, and Surya (2017) have shown experimentally that the partially ordered sets start to display the asymptotic behaviour described by Kleitman and Rothschild only around $n = 40$, where n is the number of elements. Therefore, in order to successfully apply the KR result, it seems logical to start simulations with at least $n = 40$.

4.1 Model validity

To check whether a formula is almost surely valid in models, 5000 KR models were generated. Then, a given formula was evaluated in every one of them and the number of models in which it was valid was computed.

This procedure was repeated for every formula, every modal language and every size of KR structures.

4.1.1 Model validity check

An additional experiment aimed at validating the main experiment was performed. Verbrugge (2021) introduced a 12-world canonical asymptotic Kripke model, which is supposed to resemble a countably-infinite KR model. Therefore, if a formula is valid in the asymptotic model it will be valid in an infinite KR model. A canonical model, has worlds with every possible valuation in each layer. In **GL**, no world has a reflexive relation, in **S4** all worlds have them. In **K4** there is 24-worlds, because each valuation in each layer occur once in a world without

reflexive relation and once in a world with reflexive relation.

Every formula was then evaluated in the asymptotic model. If a formula was found to be invalid in sampled models, but was valid in the asymptotic model, it would indicate mistakes in the implementation of one of the modules of this project.

4.2 Frame validity

To test if a formula is almost surely valid in frames, 500 KR frames were generated. Then the formula was evaluated in each of them and frames in which the formula was valid were counted.

In order to test frame validity, I used the probabilistic algorithm described in Section 3.8. The decision was made to test 50 valuations per frame. This results in an error probability of less than $(\frac{1}{2})^{50} \approx 9 \times 10^{-16}$. A single instance of this error results in the final count being off by one. Therefore, exhaustive validity check does not seem to be necessary.

This experiment was repeated for every formula, every modal language and every size of KR structures.

5 Results

First, we need to understand the data we gathered in the main experiment. For each simulated formula, for each language, for each structure we have six data points. Each data point is a tuple consisting of a structure size n and a number of structures in which that formula was valid. For example, formula $\Diamond\top \to \Diamond(p \wedge q)$, in **GL**, in models, has the following data points:

$$\langle 40, 524\rangle \quad \langle 48, 734\rangle$$
$$\langle 56, 947\rangle \quad \langle 64, 1132\rangle$$
$$\langle 72, 1430\rangle \quad \langle 80, 1698\rangle$$

5.1 Main experiment

Using these data points, a density plot has been created for each language and structure. An example of such graph for frames in **GL** is Figure 5.1. On each plot there are 6 categories corresponding to different sizes of simulated structures.

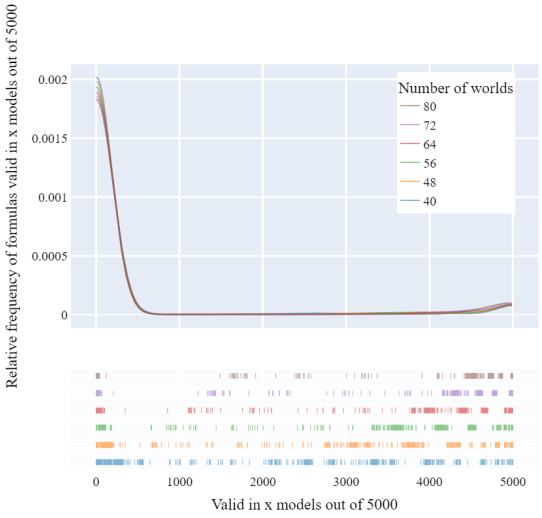


Figure 5.1: Density of formulas which are valid in x out of 5000 GL models.

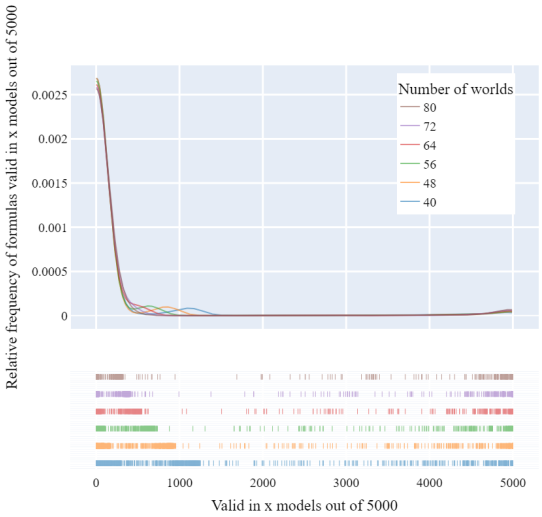


Figure 5.2: Density of formulas which are valid in x out of 5000 K4 models.

Moreover, each plot consists of two subplots. The upper one represents the relative density of formulas that were found valid in a particular number of structures. The x-axis indicates the particular number of simulated structures in which a formula is valid. For instance, if a formula is valid in all 5000 simulated models, then it will be represented as a small peak in the rightmost part of the x-axis. The peaks from all formulas are summed into the den-

sity function visible on the graph. The higher the value of the function at a location on the x-axis, the more formulas were found valid in this particular number of structures.

The bottom subplot displays each formula individually. Each formula is an individual bar placed on a location on the x-axis, corresponding to number of simulated structures in which it was valid. Each of the six layers of the subplot represent results of simulating structures of a particular size. The higher the level, the larger structures were simulated. Although one cannot see how many formulas there are at the certain position of x-axis, it is perfect for observing obscure behaviour, which disappears in the upper density plot consisting of 8047 formulas.

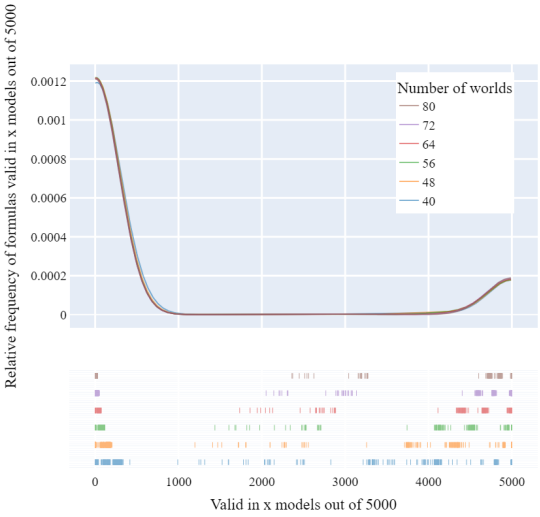


Figure 5.3: Density of formulas which are valid in x out of 5000 S4 models.

Figure 5.1 shows the results of the simulation for 5000 **GL** models. We can observe that the number of almost never valid formulas is overwhelming compared to to the number of almost surely valid formulas. On the upper density plot there are no noticeable differences between different sizes of structures. However, the bottom plot clearly shows that as the number of worlds increases, the gap between formulas approaching 0 and those approaching 5000 on the x-axis increases steadily.

Figure 5.2 summarises the results of simulating formulas in 5000 **K4** models. We can see that they are quite similar to the **GL** simulation in models. Nevertheless there are a few minor differences. First the number of almost surely valid formulas is even smaller. Secondly, the formulas are approaching validity in 0 models at a slower rate than in the case of **GL**.

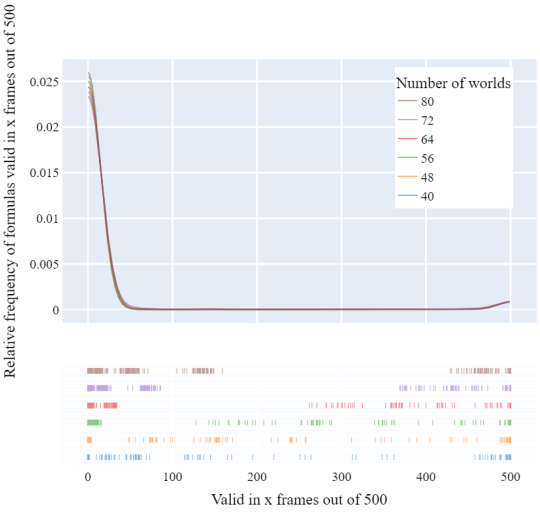


Figure 5.4: Density of formulas which are valid in x out of 500 GL frames.

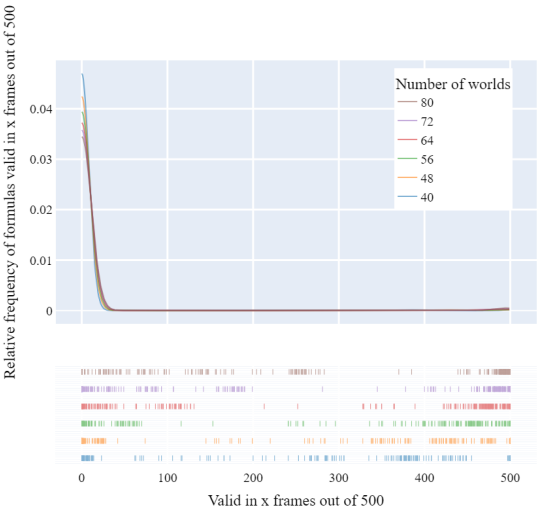


Figure 5.5: Density of formulas which are valid in x out of 500 K4 frames.

Figure 5.3 displays the results of simulating formulas in 5000 **S4** models. Again, the differences are minor compared to the previous two. They include

Logic	Tautologies	Fraction of all formulas
GL	113	1.4%
S4	917	11.4%
K4	9	0.1%

Table 5.1: Table showing a total number of tautologies in each modal language and what fraction of 8047 formulas they represent.

converging to the either side of x-axis at a faster pace, as well as containing significantly more formulas that are almost surely valid. The latter is most likely the consequence of the fact that significantly more formulas are tautologies in **S4** compared to **GL** or **K4**. This0 can be seen in Table 5.1.

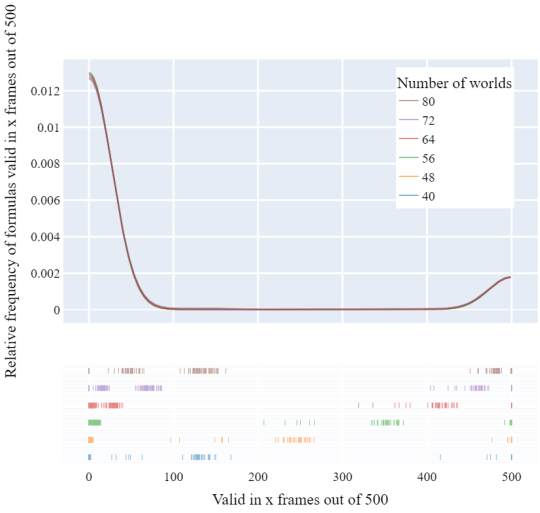


Figure 5.6: Density of formulas which are valid in x out of 500 S4 frames.

Figures 5.4, 5.5, 5.6 display the results of simulating the formulas in 500 frames of each modal language. We can see that while the upper density plots are rather similar to their model counterparts, the bottom plots differ significantly. In every language there is a group of formulas that approach the rightmost side of the x-axis as the number of worlds increases. However, now there is seemingly no second group that would approach the leftmost side of x-axis right from the beginning. Instead, there are groups of formulas, which start transitioning to the right hand side of the x-axis only when the number of worlds in simulated structures is relatively large.

These "delayed" formulas occur in all 3 modal languages. However, they do so at different pace. Whereas for **GL** and **S4** the delay is significant and only for 72 or 80 worlds this transition is noticeable, for **K4** the formulas start to migrate much sooner and at a higher pace.

5.2 Further analysis

Density graphs clearly show that vast majority of formulas are already at the extrema of the spectrum both in the case of models and in the case of frames. Moreover, the complementary subplots clearly indicate asymptotic behaviour of the other formulas. Therefore, we can assume for the following analysis that zero-one laws hold for almost sure validity in models and frames in all three logics.

In order to further analyse the results we need to classify which formulas are almost always valid and which are almost never valid. Given that the definition of zero-one laws is based on a limit, we need to capture a trend as the number of worlds approaches infinity. Therefore, to distinguish between almost surely valid formulas and their opposites, we decided to check whether the fraction of structures in which a formula is valid increases or decreases as the number of worlds grows. The importance of checking general behaviour rather than concrete values is shown in the above example where the number of models where that formula is valid is much less than the simulated 5000, but the trend is clearly increasing.

One of the ways to check the trend in a sample is linear regression. For data in the form of $\langle x, y \rangle$, such as ours, this algorithm aims to fit the line equation: $y(x) = mx + c$, where m is the gradient and c is y-intercept, to the sample. The fitting is performed by finding a line for which the sum of the squared distances between the data points and the line itself is smallest. If $m > 0$ the trend line is increasing and thus the formula is almost always valid. We had to introduce an extra check, because if a formula is always valid $m = 0$ as the trend line is a constant placed at a number of simulated structures. Therefore, if $m = 0$ and $c = 500$ for frames, or $c = 5000$ for models, then formula is also almost always valid. Otherwise a formula is treated as almost never valid.

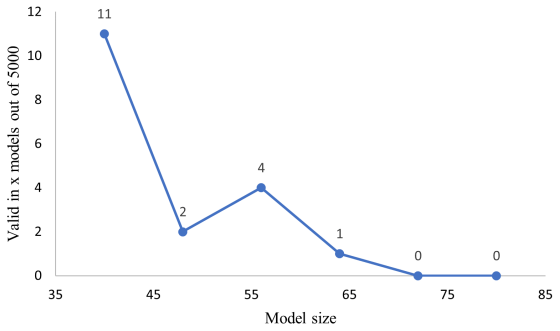


Figure 5.7: Six data points of formula $\neg(\Diamond\Diamond\neg p \wedge \neg p)$ simulated in GL, in models. It shows in how many out of 5000 randomly generated models this formula was valid for each model size.

There are other ways to check a trend, for instance by checking monotonicity. However, they are less robust than linear regression. Since our simulations are based on a sample of all possible structures it is possible that sometimes the data is not strictly monotonic. Let us take for example formula $\neg(\Diamond\Diamond\neg p \wedge \neg p)$ in **GL** in models and plot it in Figure 5.7. There we can see that although the number of models in which it is valid is decreasing it is not monotonic.

One additional analysis concerned the comparison between the number of formulas which are almost surely valid in models to those almost surely valid in frames. However, before the direct comparison the tautologies were removed from each modal language. The result can be see in Figure 5.8.

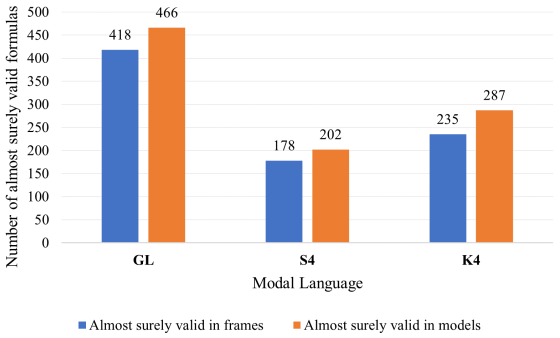


Figure 5.8: Number of almost surely valid formulas in models and frames after excluding tautologies in each language.

We can see that across all languages there are more formulas almost surely valid in models than in frames. Nevertheless the difference is relatively small, as it does not exceed 0.6 percentage point. We can see that in Table 5.2 which shows the percentages of non-tautology formulas that are almost surely valid in models and in frames.

Logic	Percentage of almost surely valid formulas in models	Percentage of almost surely valid formulas in frames
GL	5.9%	5.3%
S4	2.8%	2.5%
K4	3.6%	3.2%

Table 5.2: Table shows a percentage of non-tautology formulas that are almost surely valid in models and in frames per logic.

Furthermore, we can observe that there are significantly more almost surely valid formulas in **GL** compared to **S4** and **K4**.

Next analysis entailed verifying if a formula will be almost surely valid in **K4** if it is almost surely valid in **GL** and **S4**. This seems to be the case in general as indicated by high correlation of validity in **K4** to validities in **GL** and **S4** (see Appendix E). However, this is not a rule, because counterexamples have been found. In the case of model validity there are 2 such formulas and in the case of frame validity there are 9 of them. They are shown in Appendix F.

5.3 Asymptotic model experiment

This experiment was supposed to validate the methods used in this project. For every language, validity of every formula was checked on the asymptotic model and compared to the empirically determined almost sure model validity. For each language an adjusted asymptotic model was used, to reflect the constraints of that language. Specific adjustments are given in Section 4.1.1

After checking all formulas, the set of formulas valid in asymptotic model is the same as the set of formulas empirically determined to be almost surely valid in models in **GL** and **S4**.

However, in the case of **K4** there were 5 formulas, which were almost surely not valid according to

empirical data, but they were valid in asymptotic model. These 5 formulas can be found in the Appendix G. As an additional experiment, these 5 formulas were simulated in 5000 **K4** models containing 1000 worlds. In these enormous models, these formulas were always valid.

Therefore, this experiment confirms the correctness of our implementation.

6 Discussion

Throughout this project several tools for logical simulations and analysis were developed. Then, a set of 8047 formulas was tested in models and frames of 6 different sizes across three modal languages. Ultimately, the results were analysed primarily through a combination of density plots.

Density plots clearly indicate existence of zero-one laws for all three modal languages. In case of model validity the behaviour appears to be clear, given that all formulas approach the limit of either 0 or 1.

However, the result for frame validity is much more surprising. Although the majority of the formulas are located at the limits, we can see that some formulas only start to be almost surely valid when the number of worlds is significant.

This has several potential consequences. First, given that we have modeled structures of up to 80 worlds, there can be formulas which are almost surely valid but their "delay" is even more significant, requiring the number of worlds reaching hundreds. This entails that we can never empirically determine whether a formula is almost surely invalid, because it might be the case that it needs larger structure size to begin the transition. However, once a formula starts to approach the limit of 1 we can be certain that it is almost surely valid.

Although this behaviour is very prominent for frame validity, the experiment with asymptotic model in **K4** shows that there can be formulas, which need a significant structure size to exhibit that they are almost surely valid in models.

A possible explanation for this behaviour being prevalent in frames but not in models is the following. To check frame validity, we need to de facto check 50 models and if even one of them is invalid then the formula is not valid on that frame. Therefore, if a formula is not almost surely valid in most models, there is a high chance that it will be invalid in at least one of the 50 valuations on that frame. Hence, such formula will be found valid in no frames even in small structures. This is not the case for models, because there can be some arrangements of relations where almost surely invalid formula will be valid. Therefore, for small structures where chance of this happening is high there are formulas in the middle of the spectrum. Only with the increasing size of structures these formulas will tend to limit of 0.

Now moving to the additional analyses, Verbrugge (2021) stated that there are significantly more formulas which are almost surely valid in models than in frames. This is true that number of such formulas is higher, the difference is relatively small.

Another interesting result came from checking whether almost sure validity in **GL** and **S4** implies almost sure validity in **K4**. This seems to be true for the most formulas but it does not always hold. As a consequence, it appears that if a formula is almost surely true in both reflexive and irreflexive structures it is not necessarily almost surely true in all structures, where reflexivity is not defined.

Ultimately, it is useful to notice that these results can be extended to Grzegorczyk logic and its weak counterpart.

6.1 Limitations

The most significant limitation of this project were numbers. The correctness of the results could definitely be improved by increasing the number of valuations checked per frame, as well as number of frames and models verified per formula. Moreover, more diverse and larger sizes of simulated structures would allow to investigate the "delay" of formula transitions. Furthermore, increasing the number of formulas simulated would provide better distributions of almost surely valid formulas and their opposites.

Another limitation is human error. Although the code has been checked thoroughly, tested on a number of verified examples and verified with an additional experiment it is always possible that there is a bug in the code influencing the results. However, given the precautions even if there is some, its influence on the experiment is minimal.

6.2 Future research

This project relies on several assumptions, most important concern the imposed properties of structures. Although they were selected carefully, and their impact on a project should be limited to increasing the pace of the formulas' transition, it would be beneficial to perform the same experiments using structures generated using completely random KR framework.

Further research could also investigate what is the maximum delay of the formulas undergoing the transition and perhaps axiomatize the set of such formulas.

References

Clarke, E. M., & Schlingloff, H. (2001). Model checking. *Communications of the ACM*, *52*, 74 - 84.

Compton, K. J. (1988, August). The computational complexity of asymptotic problems I: Partial orders. *Information and Computation*, *78*(2), 108–123. doi: 10.1016/0890-5401(88)90032-6

Ebbinghaus, H., & Flum, J. (2005). *Finite model theory: Second edition.* Springer Berlin Heidelberg.

Fagin, R. (1976). Probabilities on finite models. *Journal of Symbolic Logic*, *41*(1), 50–58. doi: 10.1017/S0022481200051756

Garson, J. (2021). Modal Logic. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2021 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2021/entries/logic-modal/`.

Glebskii, Y. V., Kogan, D. I., Liogon'kii, M. I., & Talanov, V. A. (1972). Range and degree of realizability of formulas in the restricted predicate calculus. *Cybernetics*, *5*(2), 142–154. doi: 10.1007/bf01071084

Grädel, E., Kolaitis, P., Libkin, L., Marx, M., Spencer, J., Vardi, M., ... Weinstein, S. (2007). *Finite model theory and its applications.* Springer Berlin Heidelberg.

Halpern, J. Y. (2006). From statistical knowledge bases to degrees of belief. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems - PODS '06.* ACM Press. doi: 10.1145/1142351.1142367

Halpern, J. Y., & Kapron, B. M. (1994). Zero-one laws for modal logic. *Annals of Pure and Applied Logic*, *69*, 157–193.

Henson, J., Rideout, D., Sorkin, R. D., & Surya, S. (2017). Onset of the asymptotic regime for (uniformly random) finite orders. *Experimental Mathematics*, *26*(3), 253-266. doi: 10.1080/10586458.2016.1158134

Hodges, W. (2022). Model Theory. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2022 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/spr2022/entries/model-theory/`.

Immerman, N. (2012). *Descriptive complexity.* Springer New York.

Kaufmann, M., & Kaufmann, M. (1988). A counterexample to the 0-1 law for existential monadic second-order logic. In *CLI Internal Note 32, Computational Logic Inc.*

Kleitman, D. J., & Rothschild, B. L. (1975). Asymptotic enumeration of partial orders on a finite set. *Transactions of the American Mathematical Society*, *205*, 205–220.

Lagniez, J.-M., Le Berre, D., de Lima, T., & Montmirail, V. (2016, July). On Checking Kripke Models for Modal Logic K. In *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning co-located with International Joint Conference on Automated Reasoning (IJCAR 2016), Coimbra, Portugal, July 2nd, 2016.* Coimbra, Portugal.

Le Bars, J.-M. (2002). The 0-1 law fails for frame satisfiability of propositional modal logic. In *Proceedings 17th annual IEEE symposium on logic in computer science* (p. 225-234). doi: 10.1109/LICS.2002.1029831

Lin, T. (2020, May). *Benchmark of popular graph/network packages v2.* Timothy Lin. Retrieved from `https://www.timlrx.com/blog/benchmark-of-popular-graph-network-packages-v2`

Moura, D. (2021, Jul). *R vs. python vs. julia.* Towards Data Science. Retrieved from `https://towardsdatascience.com/r-vs-python-vs-julia-90456a2bcbab`

Priest, G. (2008). *An introduction to non-classical logic: From if to is* (2nd ed.).

Cambridge University Press. doi: 10.1017/CBO9780511801174

Prokhorov, A. V. (2011). Zero-one law. *Encyclopedia of Mathematics*. Retrieved from `https://encyclopediaofmath.org/index.php?title=Zero-one\_law&oldid=11252`

Schwarz, W. (2022, Jun). *Wo/tpg: Tree proof generator*. Github. Retrieved from `https://github.com/wo/tpg`

Van Benthem, J. (1984). Correspondence theory. In D. Gabbay & F. Guenthner (Eds.), *Handbook of philosophical logic: Volume II: Extensions of classical logic* (pp. 167–247). Dordrecht: Springer Netherlands. doi: 10.1007/978-94-009-6259-0_4

Van Gelder, J. (2021). *A twitter bot based on a tableau solver for GL logic* [BSc AI thesis]. University of Groningen.

Van Loo, T. (2017). *A tableau prover for GL provability logic.* [BSc AI thesis]. University of Groningen.

Verbrugge, R. (n.d.). *Zero-one laws for provability logic and some of its siblings.*

Verbrugge, R. (2017). Provability Logic. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2017 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2017/entries/logic-provability/`.

Verbrugge, R. (2021). Zero-one laws for provability logic: Axiomatizing validity in almost all models and almost all frames. In L. Libkin (Ed.), *Proceedings of the 36th annual ACM/IEEE symposium on logic in computer science.*

Wallis, W. (2013). *A beginner's guide to discrete mathematics.* Birkhäuser Boston.

Wikipedia contributors. (2022). *List of model checking tools — Wikipedia, the free encyclopedia.* `https://en.wikipedia.org/w/index.php?title=List_of_model_checking_tools&oldid=1078214549`. ([Online; accessed 13-July-2022])

A Binary tree visualization

The following figure is an example of how a formula is represented within the program.

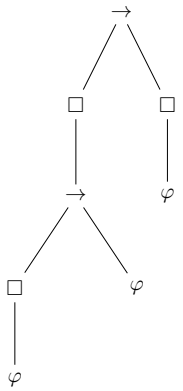


Figure A.1: Axiom of GL $\Box(\Box\varphi \to \varphi) \to \Box\varphi$ represented in a form of binary tree.

B Symbols accepted by parser

The table below shows which symbols are accepted by the parser and as which connectives they are interpreted. The program uses the symbols on the left as the internal representations of the connectives.

‘⊥’	⇐	‘⊥’ | ‘F’
‘⊤’	⇐	‘⊤’ | ‘T’
‘◊’	⇐	‘◊’ | ‘d’
‘□’	⇐	‘□’ | ‘b’
‘¬’	⇐	‘¬’ | ‘∼’
‘∧’	⇐	‘^’ | ‘∧’ | ‘&’
‘∨’	⇐	‘v’ | ‘V’ | ‘|’ | ‘∨’
‘→’	⇐	‘−>’ | ‘→’ | ‘⊃’
‘↔’	⇐	‘<>’ | ‘↔’ | ‘≡’ | ‘=’

Figure B.1: Symbols accepted by parser together with their meanings.

C BNF grammar of formulas accepted by the parser

For a formula to be accepted by the parser and converted to a binary tree it has to follow the grammar described below.

```
<atom>          ::=  '⊤' | '⊥' | <identifier> | '(' <formula> ')'

<literal>       ::=  <atom> {'¬' <atom> | '□' <atom> | '◊' <atom> }

<conjunction>   ::=  <literal> {'∨' <literal> }

<disjunction>   ::=  <conjunction> {'∧' <conjunction> }

<implication>   ::=  <disjunction> ['→' <disjunction> ]

<formula>       ::=  <implication> ['↔' <implication> ]
```

Figure C.1: Grammar of formulas accepted by a parser in Backus Normal Form.

D Selected formulas

This is a list of 47 selected formulas used in the experiment. The list is split into segments based on the reason for the inclusion of a formula in this list.

Axiom of model and frame validity in **GL**:

$\Box\Box\Box\bot$

The remaining axioms of model validity in **GL**:

$\Diamond\top \to \Diamond(p \wedge q)$
$\Diamond\top \to \Diamond(p \wedge \neg q)$
$\Diamond\top \to \Diamond(\neg p \wedge q)$
$\Diamond\top \to \Diamond(\neg p \wedge \neg q)$
$\Diamond\Diamond\top \to \Diamond(p \wedge (q \wedge \Diamond(p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(p \wedge (q \wedge \Diamond(p \wedge \neg q)))$
$\Diamond\Diamond\top \to \Diamond(p \wedge (q \wedge \Diamond(\neg p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(p \wedge (q \wedge \Diamond(\neg p \wedge \neg q)))$
$\Diamond\Diamond\top \to \Diamond(p \wedge (\neg q \wedge \Diamond(p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(p \wedge (\neg q \wedge \Diamond(p \wedge \neg q)))$
$\Diamond\Diamond\top \to \Diamond(p \wedge (\neg q \wedge \Diamond(\neg p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(p \wedge (\neg q \wedge \Diamond(\neg p \wedge \neg q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (q \wedge \Diamond(p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (q \wedge \Diamond(p \wedge \neg q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (q \wedge \Diamond(\neg p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (q \wedge \Diamond(\neg p \wedge \neg q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (\neg q \wedge \Diamond(p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (\neg q \wedge \Diamond(p \wedge \neg q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (\neg q \wedge \Diamond(\neg p \wedge q)))$
$\Diamond\Diamond\top \to \Diamond(\neg p \wedge (\neg q \wedge \Diamond(\neg p \wedge \neg q)))$

Examples of axioms taken from Halpern and Kapron (1994):

$\neg(p \wedge \Diamond(\neg p \wedge \Diamond(p \wedge \Diamond\neg p)))$
$\neg(p \wedge \Diamond(\neg p \wedge (q \wedge \Diamond(\neg q \wedge (p \wedge \Diamond\neg p)))))$
$\neg(p \wedge \Diamond(\neg p \wedge (\neg q \wedge \Diamond(q \wedge (p \wedge \Diamond\neg p)))))$
$(p \wedge \Diamond(\neg p \wedge ((p \to q) \wedge \Diamond\neg(p \to q)))) \to \Diamond(p \wedge \Diamond q)$
$(p \wedge \Diamond(\neg p \wedge ((p \leftrightarrow q) \wedge \Diamond\neg(p \leftrightarrow q)))) \to \Diamond(p \wedge \Diamond q)$
$(p \wedge \Diamond(\neg p \wedge ((p \to q) \wedge \Diamond\neg(p \to q)))) \to \Diamond(\neg p \wedge \Diamond q)$
$(p \wedge \Diamond(\neg p \wedge ((p \leftrightarrow q) \wedge \Diamond\neg(p \leftrightarrow q)))) \to \Diamond(\neg p \wedge \Diamond q)$
$(p \wedge \Diamond(\neg p \wedge ((p \to q) \wedge \Diamond\neg(p \to q)))) \to \Diamond(p \wedge \Diamond\neg q)$
$(p \wedge \Diamond(\neg p \wedge ((p \leftrightarrow q) \wedge \Diamond\neg(p \leftrightarrow q)))) \to \Diamond(p \wedge \Diamond\neg q)$
$(p \wedge \Diamond(\neg p \wedge ((p \to q) \wedge \Diamond\neg(p \to q)))) \to \Diamond(\neg p \wedge \Diamond\neg q)$
$(p \wedge \Diamond(\neg p \wedge ((p \leftrightarrow q) \wedge \Diamond\neg(p \leftrightarrow q)))) \to \Diamond(\neg p \wedge \Diamond\neg q)$
$(p \to \Box p) \vee \Diamond p$
$(p \to \Box p) \vee \Diamond(p \wedge q)$
$(\neg p \to \Box\neg p) \vee \Diamond p$
$(p \to \Box p) \vee \Diamond(p \wedge q)$
$(\neg p \to \Box\neg p) \vee \Diamond(p \wedge q)$

Formulas used to disprove zero-one laws for modal logic **K**:

$q \wedge \neg p \wedge \Box\Box((p \vee q) \to \neg\Diamond(p \vee q)) \wedge \Box\Diamond p$
$\neg\Box\Box(p \leftrightarrow \neg\Diamond p)$

Example axioms of frame validity in **GL**:

$(\Diamond\Diamond\top \wedge (\Diamond(\Diamond\top \wedge \Box p) \wedge \Diamond(\Diamond\top \wedge \Box q))) \to \Box(\Diamond\top \to \Diamond(p \wedge q))$
$(\Diamond\Diamond\top \wedge (\Diamond(\Diamond\top \wedge \Box p) \wedge \Diamond(\Diamond\top \wedge \Box\neg q))) \to \Box(\Diamond\top \to \Diamond(p \wedge \neg q))$
$(\Diamond\Diamond\top \wedge (\Diamond(\Diamond\top \wedge \Box\neg p) \wedge \Diamond(\Diamond\top \wedge \Box q))) \to \Box(\Diamond\top \to \Diamond(\neg p \wedge q))$
$(\Diamond\Diamond\top \wedge (\Diamond(\Diamond\top \wedge \Box\neg p) \wedge \Diamond(\Diamond\top \wedge \Box\neg q))) \to \Box(\Diamond\top \to \Diamond(\neg p \wedge \neg q))$
$(\Diamond\Diamond\top \wedge (\Diamond(\Box\bot \wedge p) \wedge \Diamond(\Box\bot \wedge q))) \to \Diamond(\Diamond p \wedge \Diamond q)$
$(\Diamond\Diamond\top \wedge (\Diamond(\Box\bot \wedge p) \wedge \Diamond(\Box\bot \wedge \neg q))) \to \Diamond(\Diamond p \wedge \Diamond\neg q)$
$(\Diamond\Diamond\top \wedge (\Diamond(\Box\bot \wedge \neg p) \wedge \Diamond(\Box\bot \wedge q))) \to \Diamond(\Diamond\neg p \wedge \Diamond q)$
$(\Diamond\Diamond\top \wedge (\Diamond(\Box\bot \wedge \neg p) \wedge \Diamond(\Box\bot \wedge \neg q))) \to \Diamond(\Diamond\neg p \wedge \Diamond\neg q)$

E Pearson correlation analysis of the generated dataset

The following correlation matrix consists of the following variables: tautology in **GL**, contradiction in **GL**, tautology in **S4**, contradiction in **S4**, tautology in **K4**, contradiction in **K4**, formula depth, almost sure model validity in **GL**, almost sure frame validity in **GL**, almost sure model validity in **S4**, almost sure frame validity in **S4**, almost sure model validity in **K4**, almost sure frame validity in **K4**.

We can observe that being a tautology in **GL** is highly correlated with its almost sure validity in **GL** in both models and frames. The same can be said about tautologies in **S4** and almost sure validity in **S4**, but the correlation is much higher. Additionally, tautologies in **S4** and **GL** are moderately correlated with almost sure validity in **K4** in both models and frames. Furthermore, tautologies in **K4** seem to be barely correlated to almost sure validities in that language. However, that is a due to there being only 9 tautologies in **K4** in the dataset.

Moreover, almost sure validity in models is extremely highly correlated to its frame counterpart in all 3 logics. Again, we can see that almost sure validity in **GL** and **S4** is highly correlated with almost sure validity in , almost sure model validity in **GL**, almost sure frame validity in **K4**.

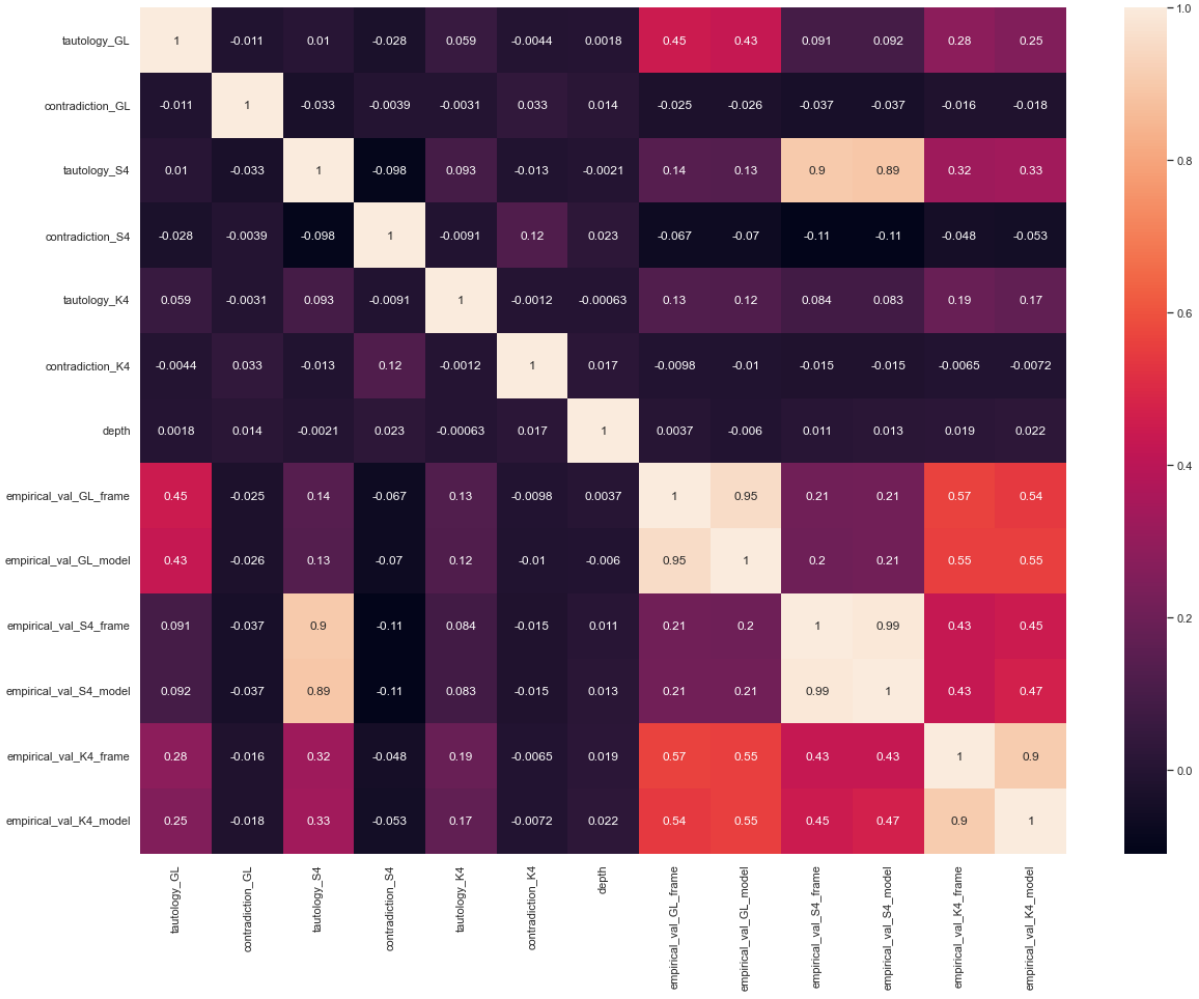


Figure E.1: Pearson correlation matrix

F Formulas almost surely valid in GL and S4, but not in K4

The following is the list of formulas which were found to be almost surely valid in models of **GL** (transitive, irreflexive) and **S4** (transitive, reflexive), but they were not almost surely valid in models of **K4** (transitive).

```
1 ( ◇ p ∧ ( ( ◇ □ q → ◇ ( p ∧ ( ( ( p ∧ q ) ∨ ¬ p ) ∧ ◇ ◇ p ) ) ) → ( ( ( q ∨ ( □ q ↔ p ) ) → ( ( q ∨ ( ◇ ⊤ ∨ ( ¬ p ↔ ¬ q
) ) ) ∧ ◇ q ) ) ∧ ( □ p ∧ ( ( ◇ ¬ p ∨ ◇ ( q ↔ p ) ) ∧ q ) ) ) ) ) ∨ ¬ ( ◇ □ □ ( ¬ p ∧ ◇ p ) ∧ ( ( □ ( ¬ q ∨ ( p → q ) ) →
◇ p ) ∧ ( ¬ p ∧ ¬ q ) ) )
2 ¬ ( ◇ ⊤ ↔ ( □ ⊥ ↔ ( ( ¬ p ∨ □ ⊥ ) ↔ ◇ ( q → ¬ p ) ) ) ) ∨ □ □ □ □ ( ◇ ⊤ ∨ ¬ ( q ∧ p ) )
```

Figure F.1: Formulas almost surely valid in models of GL and S4, but not in K4.

The following is the list of formulas which were found to be almost surely valid in frames of **GL** (transitive, irreflexive) and **S4** (transitive, reflexive), but they were not almost surely valid in frames of **K4** (transitive).

```
1 ( ◇ p ∧ ( ( ◇ □ q → ◇ ( p ∧ ( ( ( p ∧ q ) ∨ ¬ p ) ∧ ◇ ◇ p ) ) ) → ( ( ( q ∨ ( □ q ↔ p ) ) → ( ( q ∨ ( ◇ ⊤ ∨ ( ¬ p ↔ ¬ q
) ) ) ∧ ◇ q ) ) ∧ ( □ p ∧ ( ( ◇ ¬ p ∨ ◇ ( q ↔ p ) ) ∧ q ) ) ) ) ∨ ¬ ( ◇ □ □ ( ¬ p ∧ ◇ p ) ∧ ( ( □ ( ¬ q ∨ ( p → q ) ) →
◇ p ) ∧ ( ¬ p ∧ ¬ q ) ) )
2 ¬ ( ◇ ⊤ ↔ ( □ ⊥ ↔ ( ( ¬ p ∨ □ ⊥ ) ↔ ◇ ( q → ¬ p ) ) ) ) ∨ □ □ □ □ ( ◇ ⊤ ∨ ¬ ( q ∧ p ) )
3 ( ( ◇ ( p ↔ □ p ) ∧ ( □ □ ◇ ¬ q ∧ ( q → ◇ ⊤ ) ) ) → □ ⊥ ) ∨ ( q → □ ( ( ( ¬ q → p ) ∧ ◇ □ ⊥ ) ↔ ◇ ⊤ ) )
4 ◇ ( ( □ □ q ∨ ( ( ( □ □ ◇ q → ( ◇ p ∨ ¬ q ) ) ↔ ( ¬ p ∨ ( ◇ p → ¬ p ) ) ) ∨ □ ⊥ ) ) ∧ p ) ∨ □ ( ( ◇ ⊤ ∨ q ) → ◇ ( p ∨
¬ ( q ∧ □ ⊥ ) ) )
5 □ □ ( ◇ ( ( ◇ ¬ ( ¬ q ∨ ◇ q ) ∨ q ) ∧ p ) ∨ ( ( ◇ ( ¬ p ∧ □ ⊥ ) ∧ ¬ p ) → □ ⊥ ) )
6 ◇ □ ⊥ → ( ◇ ◇ □ ◇ ( ( p ↔ q ) ∧ ◇ q ) ∨ ¬ ( □ ¬ ( □ ( ¬ q ∧ □ ⊥ ) ↔ ( ◇ □ ◇ p ↔ □ q ) ) → ¬ ( ( ◇ ◇ ◇ ¬ q ∧ ( p →
◇ p ) ) → □ ¬ ( q ∨ p ) ) ) )
7 □ ( ( ◇ ⊤ ∧ ◇ ( q ∨ ( ( ¬ p ∧ ( ( p → q ) ∨ □ ⊥ ) ) ∨ ◇ ¬ q ) ) ) ∨ q ) ∨ ◇ ¬ ( ¬ ( ¬ p ∧ ◇ ⊤ ) ↔ ( ( ¬ p ↔ ( q ∧ ( □ ¬
p ∧ □ p ) ) ) ∨ ◇ □ ⊥ ) )
8 ◇ ( ( q ∨ ( ( p ∨ ◇ ¬ p ) → ◇ q ) ) ∧ p ) ∨ ( p ∨ □ ( □ ◇ ( ( ¬ ( q ↔ p ) ↔ p ) ∨ ( q ∨ □ ( ( p ∧ q ) → ◇ ¬ p ) ) ) ∨ (
q ↔ ( ( ( p ∧ ( ¬ q → q ) ) → ( ( q ∧ p ) ∨ p ) ) ↔ □ ⊥ ) ) ) )
9 ( ◇ ⊤ ↔ ◇ ◇ p ) ∨ ( ( ¬ ( ( p ↔ ( ( q ∧ ( ◇ q ∨ q ) ) ↔ ( ( p → ( ¬ q ∧ ¬ p ) ) ↔ ( q ↔ ( ¬ q ∨ ◇ ¬ p ) ) ) ) ) ) → ◇ ⊤ ) ∨
◇ ⊤ ) ↔ ( ¬ ( □ □ ¬ ( q → □ ⊥ ) ∧ ◇ ¬ p ) ↔ ( ( p ∨ ◇ ¬ ( p ↔ q ) ) → ( p ∧ ( ¬ ( ( q ↔ ( ◇ p ∨ □ ( q ↔ p ) ) ) ∨ ¬ q ) ∧
□ ¬ q ) ) ) ) )
```

Figure F.2: Formulas almost surely valid in frames of GL and S4, but not in K4.

G Formulas valid in asymptotic model, but invalid according to empirical data

This is a list of 5 formulas which were found invalid in all models consisting of 80 worlds or less. However, when tested in larger models (1000 worlds) they were always valid. This suggests that for this formulas to display the asymptotic behaviour large structures are needed.

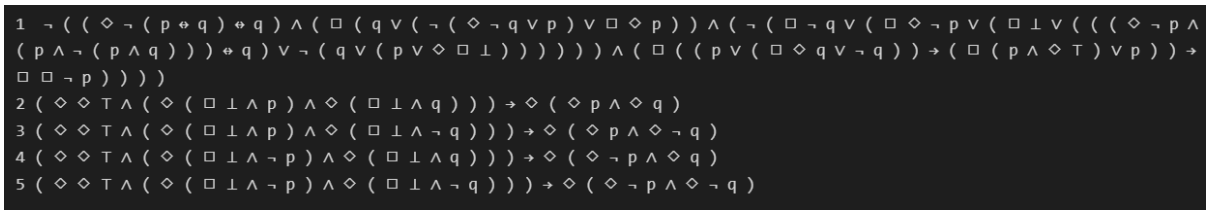


Figure G.1: 5 formulas which were found invalid in the main experiment, but were valid in the asymptotic model.