Studies in Logic, Vol. 17, No. 3 (2024): 86–101
PII: 1674-3202(2024)-03-0086-16

Decidability for Modal Logic with Counting ML(#) in Different Frame Classes*

Xiaoxuan Fu Zhiguang Zhao

Abstract. In the present paper, we give the decision procedure of satisfiability of modal logic with counting ML(#) in different frame classes, by two types of methods, one by modifying the decision algorithm of satisfiability for ML(#) with respect to the class of all Kripke frames as described by J. van Benthem and T. Icard (2021), the other by reducing decidability of ML(#) to that of basic modal logic. We also show the decidability of graded modal logic with counting GML(#) with respect to the class of all Kripke frames.

1 Introduction

In the literature, there are many works combining cardinality comparison with logical languages ([1, 3, 6, 7]). In particular, in [2], van Benthem and Icard investigated modal logic with counting ML(#), and studied its model-theoretic properties: some invariance results and finite depth property of ML(#) are proved, and it was also shown that ML(#) is decidable via a normal form argument. In this paper, we study the decidability of ML(#) in different frame classes in two different ways: the first is still via the normal form argument, but with modified algorithms; the second is via reducing ML(#)-formulas to basic modal formulas. We also show the decidability of graded modal logic with counting GML(#) with respect to the class of all Kripke frames.

The structure of the paper is as follows: Section 2 gives preliminaries on ML(#). Section 3 recalls the decision algorithm for ML(#) in the class of all Kripke frames. Section 4 gives the decision algorithms for ML(#) in different frame classes. Section 5 shows the decidability of graded modal logic with counting GML(#) with respect to the class of all Kripke frames.

Received 2023-02-23

Xiaoxuan Fu School of Humanities, China University of Political Science and Law
xfuuva@gmail.com

Zhiguang Zhao School of Mathematics and Statistics, Taishan University
zhaozhiguang23@gmail.com

*The research of the first author is supported by Tsinghua University Initiative Scientific Research Program. The research of the second author is supported by Taishan Young Scholars Program of the Government of Shandong Province, China (No. tsqn201909151), Shandong Provincial Natural Science Foundation, China (No. ZR2023QF021) and the Support Plan on Science and Technology for Youth Innovation of Universities in Shandong Province (No. 2021KJ086).

2 Modal Logic with Counting ML(#)

In the present section, we give preliminaries on the modal language with counting $\mathrm{ML}(\#)$. For more details, see [2, Section 7].

Syntax. Given a set Prop of propositional variables, we define the formulas and numerical terms of $\mathrm{ML}(\#)$ as follows:

$$\begin{array}{rl} \text{formulas:} & p \mid \bot \mid \top \mid \neg\varphi \mid \varphi \wedge \psi \mid \#\varphi \succsim \#\psi \\ \text{numerical terms:} & \#\varphi \end{array}$$

where $p \in \mathrm{Prop}$. We use standard abbreviations for $\vee, \rightarrow, \leftrightarrow$. In addition, we define the following abbreviations:

- $\#\varphi \succ \#\psi$ is defined as $(\#\varphi \succsim \#\psi) \wedge \neg(\#\psi \succsim \#\varphi)$;
- $\#\varphi = \#\psi$ is defined as $(\#\varphi \succsim \#\psi) \wedge (\#\psi \succsim \#\varphi)$;
- The standard modality $\Diamond\varphi$ is defined as $\#\varphi \succ \#\bot$;
- $\Box\varphi$ is defined as $\neg\Diamond\neg\varphi$.

Definition 1 (Counting depth)**.** The counting depth of an arbitrary formula is defined recursively as follows:

- $d(p) = d(\bot) = d(\top) = 0$;
- $d(\neg\varphi) = d(\varphi)$;
- $d(\varphi \wedge \psi) = max\{d(\varphi), d(\psi)\}$;
- $d(\#\varphi \succsim \#\psi) = max\{d(\varphi), d(\psi)\} + 1$.

Semantics. $\mathrm{ML}(\#)$-formulas are interpreted on Kripke frames $\mathbb{F} = (W, R)$ where $W \neq \varnothing$ is the domain and R is a binary relation on W. A Kripke model is a tuple $\mathbb{M} = (\mathbb{F}, V)$ where $V : \mathrm{Prop} \rightarrow \mathrm{P}(W)$ is a valuation on W. We use $R[s] = \{t : Rst\}$ to denote the set of successors of s, and $[\![\varphi]\!]^{\mathbb{M}} = \{w \in W \mid \mathbb{M}, w \Vdash \varphi\}$ to denote the truth set of φ in $\mathbb{M}$.

The satisfaction relation for the basic case and Boolean connectives are defined as usual. For numerical terms,

$$[\![\#\varphi]\!]^{\mathbb{M},s} = |R[s] \cap [\![\varphi]\!]^{\mathbb{M}}|,$$

i.e. $[\![\#\varphi]\!]^{\mathbb{M},s}$ is the number of successors of s where φ is true.

For cardinality comparison formulas,

$$\mathbb{M}, s \Vdash \#\varphi \succsim \#\psi \text{ iff } [\![\#\varphi]\!]^{\mathbb{M},s} \geq [\![\#\psi]\!]^{\mathbb{M},s}$$

i.e. $\#\varphi \succsim \#\psi$ is true at s if more (or the same number of) R-successors of s make φ true than make ψ true.

3 Decision Algorithm for Satisfiability with Respect to all Kripke Frames

In this section, we recall the decision algorithm for ML(#)-formulas with respect to the class of all Kripke frames in [2, Section 7], by the use of normal forms. This section is about existing results (with some more detailed proofs), we repeat it only for the sake of making the proofs in the later sections easier.

In this paper, we fix a finite number of propositional variables $p_1, \ldots, p_m$.

Definition 2. Given m propositional variables $p_1, \ldots, p_m$, a literal is a formula of the form p_i or $\neg p_i$, for $i = 1, \ldots, m$. A complete conjunctive clause is a formula of the form $(\neg)p_1 \wedge \ldots \wedge (\neg)p_m$, i.e. a conjunction of m literals of different propositional variables.

Definition 3 (n-type, see Definition 4 in [2]). Fix m propositional variables $p_1, \ldots, p_m$, the n-types are defined inductively on n:

- A 0-type is a complete conjunctive clause;
- An $(n+1)$-type is a conjunction of a 0-type and a complete set of inequalities which form a linear order (i.e. reflexive, transitive and total, but not necessarily anti-symmetric)

$$\#T_{1,1} = \#T_{1,2} = \ldots = \#T_{1,k_1} \succ \#T_{2,1} = \#T_{2,2} = \ldots = \#T_{2,k_2} \succ$$

$$\ldots \succ \#T_{t,1} = \#T_{t,2} = \ldots = \#T_{t,k_t}$$

  where $T_{1,1}, T_{1,2}, \ldots, T_{t,k_t}$ is a complete list of all formulas that are disjunctions (possibly an empty disjunction) of n-types.

Example 1. Given $m = 1$ and propositional variable p_1, the formulas p_1 and $\neg p_1$ are all 0-types, and $p_1 \wedge \#(p_1 \vee \neg p_1) \succ \#p_1 = \#\neg p_1 \succ \#\bot$ is a 1-type.

Notice that n-types can be unsatisfiable, e.g. the 1-type $p_1 \wedge \#(p_1 \vee \neg p_1) \succ \#p_1 = \#\bot \succ \#\neg p_1$, since the number of successors satisfying $\bot$ could not be larger than 0.

We can see that the inductive step in Definition 3 makes sense because it is easy to show inductively that the set of n-types is finite for each n (when fixing m propositional variables):

Proposition 1. *Fix m propositional variables $p_1, \ldots, p_m$, there are finitely many n-types for each n.*

Proof. For the case $n = 0$, there are 2^m many different complete conjunctive clauses.

Suppose that for $n = k$ we have finitely many different k-types, then for $n = k + 1$, there are finitely many possible disjunctions of k-types, therefore there are finitely many possible linear orders of those disjunctions. Together with the fact that there are 2^m many 0-types, we have that there are finitely many $(k+1)$-types. $\square$

Proposition 2 (Fact 6 in [2], with slight revision)**.** *Each formula φ of ML$(\#)$ with counting depth at most n is equivalent to a disjunction of n-types, and this disjunction can be computed by an algorithm.*

Proof. We prove this by induction on n.

For the case $n = 0$, φ is of depth 0, so it is a propositional formula, therefore it can be written in disjunctive normal form, so it can be written as a disjunction of complete conjunctive clauses, i.e. 0-types. This can be done by an algorithm.

For the case $n = k+1$, every formula of depth at most $k+1$ can be rewritten into a Boolean combination of propositional variables and formulas $\#\psi \succsim \#\theta$ with ψ, θ of counting depth at most k. By induction hypothesis, these ψ's and θ's can be rewritten into their respective equivalent disjunctions of k-types. Therefore, the whole formula is equivalent to a disjunction of conjunctions of such statements (without loss of generality we assume that each disjunction branch has a complete conjunctive clause as a subformula), where negations of cardinality comparison formulas can be replaced by strict inequalities. Therefore, some comparisons between disjunctions of k-types are already given, and then we replace this formula by the disjunction of all completions of the comparisons to fill in comparisons between all disjunctions of k-types, which is possible by the linearity of $\succsim$. All of these can be achieved by an algorithm. $\square$

Proposition 3. *For any two different n-types φ_n and ψ_n, the formula $\varphi_n \wedge \psi_n$ is not satisfiable.*

Proof. For the case $n = 0$, it is obvious.

For the case $n = k+1$, for any two different n-types φ_{k+1} and ψ_{k+1}, if their 0-type parts are not the same, then their conjunction is not satisfiable. If their 0-type parts are the same, then their linear order must be different, so there are two disjunctions of k-types $\bigvee \alpha_i$ and $\bigvee \beta_i$ such that their relative order is different in φ_{k+1} and ψ_{k+1}, so the conjunction $\varphi_{k+1} \wedge \psi_{k+1}$ is not satisfiable. $\square$

Proposition 4. *Suppose that $\alpha_1, \dots, \alpha_t$ enumerate all the n-types, then the formula $\top \leftrightarrow \alpha_1 \vee \dots \vee \alpha_t$ is valid.*

Proof. By Proposition 2, since $d(\top) = 0 \leq n$, it is equivalent to a disjunction of n-types. Therefore, $\top \rightarrow \alpha_1 \vee \dots \vee \alpha_t$ is valid. The validity of $\alpha_1 \vee \dots \vee \alpha_t \rightarrow \top$ is trivial. $\square$

The following proposition is useful in the decision algorithm for the class of reflexive frames:

Proposition 5. *For each* $(k+1)$*-type* φ_{k+1}*, if* φ_{k+1} *is satisfiable, then we can recognize a unique* k*-type* φ_k *such that* $\varphi_{k+1} \to \varphi_k$ *is valid propositionally and for all other* k*-types* ψ_k*,* $\varphi_{k+1} \wedge \psi_k$ *is not satisfiable. Otherwise* φ_{k+1} *is not satisfiable and* $\varphi_{k+1} \to \varphi_k$ *is valid for all* k*-types* φ_k*. In both cases we can recognize such a* k*-type efficiently, which we will call the canonical* k*-type of* φ_{k+1}*.*

Proof. **Existence**:

For the case $k = 0$, since each 0-type is a complete conjunctive clause, and each 1-type is a conjunction of a 0-type and a complete set of inequalities which form a linear order of disjunctions (possibly empty) of 0-types, we can take φ_k to be the 0-type part of the 1-type.

Now for the case $k > 0$. Since each $(k+1)$-type φ_{k+1} is a conjunction of a 0-type and a complete set of inequalities which form a linear order of disjunctions (possibly empty) of k-types, and each k-type ψ_k is a conjunction of a 0-type and a complete set of inequalities which form a linear order of disjunctions (possibly empty) of $(k-1)$-types, from Proposition 2, we get that each $(k-1)$-type can be effectively rewritten as a disjunction of k-types, so we can find all such disjunctions of k-types in subformulas of φ_{k+1}, which form a sub-linear order. By taking this sub-linear order on the $(k-1)$-types together with the 0-type of φ_{k+1}, we get the required formula φ_k.

Uniqueness:

If there are two different k-types φ_k and φ'_k such that $\varphi_{k+1} \to \varphi_k$ and $\varphi_{k+1} \to \varphi'_k$ are valid, then by the satisfiability of φ_{k+1} we get the satisfiability of $\varphi_k \wedge \varphi'_k$, a contradiction.

To show that for all other k-types ψ_k, the conjunction $\varphi_{k+1} \wedge \psi_k$ is not satisfiable, suppose otherwise, $\varphi_{k+1} \wedge \psi_k$ is satisfiable, then by the validity of $\varphi_{k+1} \to \varphi_k$, we have that $\varphi_k \wedge \psi_k$ is satisfiable, a contradiction.

The otherwise part follows immediately.

Since the recognition of φ_k does not depend on the satisfiability of φ_{k+1} and can be done via an algorithm, we can recognize φ_k from φ_{k+1} anyway. $\square$

Proposition 6. *For each (*$k+1$*)-type* φ_{k+1}*, if its canonical* k*-type* φ_k *is satisfiable, then for any disjunction of* k*-type* T*, either* $\varphi_k \to T$ *is valid, or* $\varphi_k \to \neg T$ *is valid. If* φ_k *is not satisfiable, then* $\varphi_k \to T$ *is valid for all disjunction of* k*-type* T*.*

Proof. If φ_k is satisfiable, then when T contains φ_k as a disjunction branch, then clearly $\varphi_k \to T$ is valid; if T does not contain φ_k as a disjunction branch, then $\varphi_k \wedge \alpha$ is not satisfiable for all disjunction branches α of T, so $\varphi_k \to \neg\alpha$ is valid for all α, so $\varphi_k \to \bigwedge \neg\alpha$, i.e. $\varphi_k \to \neg T$, is valid. $\square$

Definition 4.

- Given a cardinality comparison formula $\#S \succsim \#T$ where S is $\alpha_1 \vee \ldots \vee \alpha_s$ and $\#T$ is $\beta_1 \vee \ldots \vee \beta_t$ and each α_i $(1 \leq i \leq s)$ and β_j $(1 \leq j \leq t)$ is a n-type, we assign the inequality

$$x_{\alpha_1} + \ldots + x_{\alpha_s} \geq x_{\beta_1} + \ldots + x_{\beta_t}$$

  to $\#S \succsim \#T$, and denote it as $\mathrm{Ineq}(\#S \succsim \#T)$. When S or T is $\bot$, then we assign 0 to its side of the inequality.
  For cardinality comparison formula $\#S \succ \#T$, we replace $\geq$ by $>$ in the inequality above.
  For cardinality comparison formula $\#S = \#T$, we replace $\geq$ by $=$ in the inequality above.
- Given a complete set of inequalities

$$\#T_{1,1} = \#T_{1,2} = \ldots = \#T_{1,k_1} \succ \#T_{2,1} = \#T_{2,2} = \ldots = \#T_{2,k_2} \succ$$
$$\ldots \succ \#T_{t,1} = \#T_{t,2} = \ldots = \#T_{t,k_t}$$

  which form the linear order part of the $(n+1)$-type φ, for each pair of different $T_{i,j}$ and $T_{k,l}$, if according to the linear order, $\#T_{i,j} \succ \#T_{k,l}$ (resp. $\#T_{i,j} = \#T_{k,l}$, $\#T_{k,l} \succ \#T_{i,j}$), then we assign the inequality $\mathrm{Ineq}(\#T_{i,j} \succ \#T_{k,l})$ (resp. $\mathrm{Ineq}(\#T_{i,j} = \#T_{k,l})$, $\mathrm{Ineq}(\#T_{k,l} \succ \#T_{i,j})$) to it. Finally, we collect all the inequalities to form a linear inequality system $\mathrm{Sys}(\varphi)$.

Proposition 7. *Given a linear inequality system, by the Fourier-Motzkin algorithm allowing infinite cardinalities, it is decidable whether this linear inequality system has a non-negative solution (possibly some variables have infinite cardinality value).*

Proof. See [2, Section 4.2]. □

Proposition 8 (Proposition 12 in [2]). *ML(#) is decidable.*

Proof. We show that the satisfiability problem for ML(#) is decidable, for each formula θ of depth n.

We first rewrite θ by an algorithm into an equivalent disjunction of n-types. If this disjunction is empty, then we output "not satisfiable". Otherwise, we run the following algorithm for each disjunction branch φ (i.e. an n-type) of the input formula θ, according to n:

- At depth $n = 0$, check whether φ is propositionally satisfiable.
- At depth $n = k + 1$, for the given $(k+1)$-type φ, we check that
  1. the atomic part for φ is satisfiable;
  2. the linear inequality system $\mathrm{Sys}(\varphi)$ has a non-negative solution;

3. for each non-zero value of variables in step 2, check the satisfiability of its corresponding k-type.

If one of the previous steps fail, then we output "φ is not satisfiable". Otherwise, we output "φ is satisfiable".

It is easy to see that each step in the algorithm above is decidable. When the disjunction is empty, then the formula is equivalent to $\perp$ and is hence not satisfiable. When the steps described fail, it is easy to see that the formula is not satisfiable. If all the steps pass through without failure, then by induction on the depth of φ, we can find a root node satisfying the atomic part of φ, and for each non-zero value in the solution of the linear inequality system, by copying and taking disjoint subtrees, we can satisfy it at any desired number of successors for the root as described by the inequalities of stage 2. Notice that by Propositions 3 and 4, each successor node can satisfy exactly one n-type. □

4 Algorithms for Other Systems

In this section, we will modify the algorithm in the previous section to get the decision method for satisfiability with respect to different frame classes.

The frame classes we will consider are the following (the names are the corresponding names for the basic modal logic systems, here we abuse the names to denote the corresponding frame classes):

- reflexive frames (which we denote as T);
- serial frames (which we denote as D);
- equivalence relations (which we denote as S5);
- transitive and Euclidean frames (which we denote as K45);
- serial, transitive and Euclidean frames (which we denote as KD45);
- frames where each node has at most one successor (which we denote as Alt_1);
- frames where each node has at most two successors (which we denote as Alt_2).

4.1 T

We show that the satisfiability problem for ML(#) with respect to the class of reflexive frames is decidable, for each formula θ of depth n.

We first rewrite θ by an algorithm into an equivalent disjunction of n-types. If this disjunction is empty, then we output "not satisfiable". Otherwise, we run the following algorithm for each disjunction branch φ (i.e. an n-type) of the input formula θ, according to n:

- At depth $n = 0$, check whether φ is propositionally satisfiable.
- At depth $n = k + 1$, for the given $(k + 1)$-type φ,

1. check that the atomic part for φ is propositionally satisfiable;
2. check that the canonical k-type φ_k of φ is T-satisfiable;
3. find out all the disjunctions of k-types S such that S is T-implied by the canonical k-type φ_k of φ (which is essentially checking T-validity for depth-k implicative formulas $\varphi_k \to S$);
4. add, to the linear inequality system $\mathrm{Sys}(\varphi)$, inequalities corresponding to $S = \alpha_1 \vee \ldots \vee \alpha_s$ saying that $x_{\alpha_1} + \ldots + x_{\alpha_s} \geq 1$, and get the linear inequality system $\mathrm{Sys}'(\varphi)$;
5. check that the linear inequality system $\mathrm{Sys}'(\varphi)$ has a non-negative solution;
6. for each non-zero value of variables in the previous step, check the T-satisfiability of its corresponding k-type.

If one of the previous steps fail, then we output "φ is not satisfiable". Otherwise, we output "φ is satisfiable".

It is easy to see that each step in the algorithm above is decidable. We only focus on that part that is different from the case of all Kripke frames.

For depth $k+1$ case, the difference of the T case from the all Kripke frames case is that the root point is reflexive, if the canonical k-type φ_k of φ is T-satisfiable (which is a necessary requirement for φ to be T-satisfiable), then all the disjunctions of k-types S T-implied by φ_k would be true at the root node, and all the rest disjunctions of k-types are not T-satisfiable together with φ_k. So when counting the number of successors satisfying S, we need to require that the root node already satisfies S for those such that $\varphi_k \to S$ is T-valid, so the requirement that $x_{\alpha_1} + \ldots + x_{\alpha_s} \geq 1$ is necessary.

4.2 D

We show that the D-satisfiability problem for ML(#) is decidable, for each formula θ of depth n.

We first rewrite θ by an algorithm into an equivalent disjunction of n-types. If this disjunction is empty, then we output "not satisfiable". Otherwise, we run the following algorithm for each disjunction branch φ (i.e. an n-type) of the input formula θ, according to n:

- At depth $n = 0$, check whether φ is propositionally satisfiable.
- At depth $n = k+1$, for the given $(k+1)$-type φ,

1. check that the atomic part for φ is propositionally satisfiable;
2. add, to the linear inequality system $\mathrm{Sys}(\varphi)$, an inequality $x_{\alpha_1} + \ldots + x_{\alpha_t} > 0$ such that $\alpha_1, \ldots, \alpha_t$ enumerates all the k-types, and get the linear inequality system $\mathrm{Sys}'(\varphi)$;

3. check that the linear inequality system $\mathrm{Sys}(\varphi)$ has a non-negative solution;
4. check that for each non-zero value of variables in the previous step, check the D-satisfiability of its corresponding k-type.

If one of the previous steps fail, then we output "φ is not satisfiable". Otherwise, we output "φ is satisfiable".

It is easy to see that each step in the algorithm above is decidable. We only focus on that part that is different from the case of all Kripke frames.

For depth $k+1$ case, the difference of the D case from the all Kripke frames case is that the root point is serial, so $\#\top$ should have value > 0, so by the equivalence that $\top \leftrightarrow \alpha_1 \vee \ldots \vee \alpha_t$, we need an additional inequality saying that $x_{\alpha_1} + \ldots + x_{\alpha_t} > 0$.

4.3 S5

For the class of equivalence relations, we have the following normal form reduction:

First of all, since when a formula φ is satisfiable on an equivalence relation frame, then by taking the generated submodel, we have that φ is satisfiable on a full relation frame, i.e. frames where $R = W \times W$. Since the class of full relation frames is a proper subclass of the class of equivalence relation frames, the satisfiable formulas of the two classes coincide. Therefore, in what follows we consider the satisfiability problem in the class of full relation frames.

We have the following proposition, which is the basis of our algorithm in this section:

Proposition 9. *Suppose that φ has a subformula of the form $\#\psi \gtrsim \#\theta$. Then $\varphi \leftrightarrow (\varphi[\top/\#\psi \gtrsim \#\theta] \wedge (\#\psi \gtrsim \#\theta)) \vee (\varphi[\bot/\#\psi \gtrsim \#\theta] \wedge \neg(\#\psi \gtrsim \#\theta))$ is valid in the class of full relation frames.*

Proof. For any pointed full relation model $(\mathbb{M}, w)$, since $\#\psi \gtrsim \#\theta$ is either globally true or globally false,

$$
\begin{array}{ll}
 & \mathbb{M}, w \Vdash \varphi \\
\text{iff} & (\mathbb{M}, w \Vdash \varphi \text{ and } \mathbb{M} \Vdash (\#\psi \gtrsim \#\theta)) \text{ or } (\mathbb{M}, w \Vdash \varphi \text{ and } \mathbb{M} \Vdash \neg(\#\psi \gtrsim \#\theta)) \\
\text{iff} & (\mathbb{M}, w \Vdash \varphi \text{ and } \mathbb{M} \Vdash (\#\psi \gtrsim \#\theta) \leftrightarrow \top \text{ and } \mathbb{M} \Vdash (\#\psi \gtrsim \#\theta)) \text{ or} \\
 & (\mathbb{M}, w \Vdash \varphi \text{ and } \mathbb{M} \Vdash (\#\psi \gtrsim \#\theta) \leftrightarrow \bot \text{ and } \mathbb{M} \Vdash \neg(\#\psi \gtrsim \#\theta)) \\
\text{iff} & (\mathbb{M}, w \Vdash \varphi[\top/\#\psi \gtrsim \#\theta] \text{ and } \mathbb{M} \Vdash \#\psi \gtrsim \#\theta) \text{ or} \\
 & (\mathbb{M}, w \Vdash \varphi[\bot/\#\psi \gtrsim \#\theta] \text{ and } \mathbb{M} \Vdash \neg(\#\psi \gtrsim \#\theta)) \\
\text{iff} & \mathbb{M}, w \Vdash (\varphi[\top/\#\psi \gtrsim \#\theta] \wedge (\#\psi \gtrsim \#\theta)) \vee (\varphi[\bot/\#\psi \gtrsim \#\theta] \wedge \neg(\#\psi \gtrsim \#\theta)).
\end{array}
$$

□

By applying the proposition above, any formula is equivalent to a depth-1 formula, and the formula can be computed by an algorithm.

Now we show that the $S5$-satisfiability problem for $\mathrm{ML}(\#)$ is decidable, for each formula θ of depth n.

We first rewrite θ by an algorithm into an equivalent depth-1 formula, and then by an algorithm into an equivalent disjunction of 1-types. If this disjunction is empty, then we output "not satisfiable". Otherwise, we run the following algorithm for each disjunction branch φ (i.e. an 1-type) of the input formula θ:

At depth $n = 1$, for the given 1-type φ,

1. check that the atomic part for φ is satisfiable;
2. find out all the disjunctions of 0-types T such that T is S5-implied by the atomic part of φ (which is essentially checking S5-validity for depth-0 implicative formulas $\varphi_0 \to T$);
3. add, to the linear inequality system $\mathrm{Sys}(\varphi)$, inequalities corresponding to $T = \alpha_1 \vee \ldots \vee \alpha_s$ saying that $x_{\alpha_1} + \ldots + x_{\alpha_s} \geq 1$, and get the linear inequality system $\mathrm{Sys}'(\varphi)$;
4. check that the linear inequality system $\mathrm{Sys}'(\varphi)$ has a non-negative solution.

If one of the previous steps fail, then we output "φ is not satisfiable". Otherwise, we output "φ is satisfiable".

It is easy to see that each step in the algorithm above is decidable. By the previous propositions, we have the soundness of the reduction to depth-1 formula and a disjunction of 1-types. Then we check whether the formula is satisfiable in a full relation frame: we first find out all the disjunctions of 0-types implied by φ_0, and require them to have at least one successor satisfying them (namely the root node), and then if the system has a solution, we just create the corresponding number of points satisfying the corresponding 0-type. Notice that all 0-types are satisfiable.

4.4 K45

First of all, we can see that for any K45-frame $\mathbb{F} = (W, R)$, for any point w in the frame, consider the generated subframe $\mathbb{F}_w = (W_w, R_w)$ generated by w, then for any v such that $R_w wv$, we have that $R_w[w] = R_w[v]$. By an easy induction, we can show that for all points v in $\mathbb{F}_w$, we have $R_w[w] = R_w[v]$. Therefore, the domain W_w is $\{w\} \cup R_w[w]$, and the relation R_w is $W \times R_w[w]$.

Therefore, there are three possibilities:

- w has no successor, i.e. $R_w[w] = \varnothing$;
- w is not reflexive and w is accessible to an equivalence cluster, i.e. $w \notin R_w[w]$ and $R_w = (\{w\} \times R_w[w]) \cup (R_w[w] \times R_w[w])$;
- w is reflexive and belongs to an equivalence relation, i.e. $w \in R_w[w]$ and $R_w = R_w[w] \times R_w[w]$.

Therefore, if w has successors, then the generated submodel by any successor v of w is a full relation model.

We have the following decision algorithm for the K45-satisfiability problem for $\mathrm{ML}(\#)$, for each formula θ of depth n.

We first rewrite θ by an algorithm into an equivalent disjunction of n-types. If this disjunction is empty, then we output "not satisfiable". Otherwise, we run the following algorithm for each disjunction branch φ (i.e. an n-type) of the input formula θ, according to n:

- At depth $n = 0$, check whether φ is propositionally satisfiable; if not, then output "φ is not satisfiable", otherwise, output "φ is satisfiable".
- At depth $n = k + 1$, for the given $(k + 1)$-type φ,
  1. check whether the atomic part for φ is propositionally satisfiable; if not, then output "φ is not satisfiable"; otherwise, go to the next step;
  2. check whether the linear order part has no $\succ$-formulas; if there is no $\succ$-formulas, then output "φ is satisfiable", otherwise go to the next step;
  3. check whether the linear order part of φ is S5-satisfiable; if it is S5-satisfiable, then output "φ is satisfiable", otherwise output "φ is not satisfiable".

It is easy to see that each step in the algorithm above is decidable. We only focus on that part that is different from the case of all Kripke frames.

For depth $k+1$ case, we check if there is no $\succ$-formulas, if not, then we can assign value 0 to all variables for k-types in the linear inequality system, so in the model, the root node has no successor. If there are $\succ$-formulas, then there are successors of the root node w, and for any $v \in R[w]$, we have $R[w] = R[v]$, and the submodel generated by $R[v]$ is a full relation model. So φ is satisfiable at w iff the atomic part of φ is satisfiable at w and the linear order part of φ is satisfiable at v, i.e. at the generated submodel generated by v, i.e. at an S5-model.

4.5 KD45

For KD45, we can see that for any KD45-frame $\mathbb{F} = (W, R)$, for any point w in the frame, consider the generated subframe $\mathbb{F}_w = (W_w, R_w)$ generated by w, then for any v such that $R_w wv$, we have that $R_w[w] = R_w[v]$. By an easy induction, we can show that for all points v in $\mathbb{F}_w$, we have $R_w[w] = R_w[v]$. Therefore, the domain W_w is $\{w\} \cup R_w[w]$, and the relation R_w is $W \times R_w[w]$.

Therefore, there are two possibilities:

- w is not reflexive and w is accessible to an equivalence cluster, i.e. $w \notin R_w[w]$ and $R_w = (\{w\} \times R_w[w]) \cup (R_w[w] \times R_w[w])$;

- w is reflexive and is belonging to an equivalence relation, i.e. $w \in R_w[w]$ and $R_w = R_w[w] \times R_w[w]$.

Therefore, the generated submodel by any successor v of w is a full relation model. The difference between the KD45 case and the K45 case is that the root node must have at least one successor node.

Now the decision algorithm for KD45-satisfiability is given as follows, for each formula θ of depth n.

We first rewrite θ by an algorithm into an equivalent disjunction of n-types. If this disjunction is empty, then we output "not satisfiable". Otherwise, we run the following algorithm for each disjunction branch φ (i.e. an n-type) of the input formula θ, according to n:

- At depth $n = 0$, check whether φ is propositionally satisfiable; if not, then output "φ is not satisfiable", otherwise, output "φ is satisfiable".
- At depth $n = k + 1$, for the given $(k + 1)$-type φ,
  1. check whether the atomic part for φ is satisfiable; if not, then output "φ is not satisfiable"; otherwise, go to the next step;
  2. check whether the linear order part has no $\succ$-formulas; if there is no $\succ$-formulas, then output "φ is not satisfiable"; otherwise, go to the next step;
  3. check whether the linear order part of φ is S5-satisfiable; if it is S5-satisfiable, then output "φ is satisfiable", otherwise output "φ is not satisfiable".

It is easy to see that each step in the algorithm above is decidable. We only focus on the part that is different from the case of K45 frames.

For depth $k + 1$ case, we need to guarantee that the root node has successor, so if there is no $\succ$-formulas, then $\#\top = \#\bot$, which means that the root node has no successor, which is not satisfiable on a KD45 frame. The rest of the proof is similar to the K45 case.

4.6 Alt$_1$ and Alt$_2$

For Alt$_1$ and Alt$_2$, we apply another approach to get the decidability of satisfiability problems, namely reducing $\succsim$-formulas to basic modal formulas.

Proposition 10. *For any frame $\mathbb{F} = (W, R)$ in Alt$_2$,*

$$\mathbb{F} \Vdash \#\varphi \succ \#\psi \leftrightarrow ((\Box\varphi \wedge \neg\Box\psi) \vee (\Diamond\varphi \wedge \neg\Diamond\psi))$$

Proof. Consider any frame $\mathbb{F} = (W, R)$ in Alt$_2$ and any valuation V on $\mathbb{F}$ and any $w \in W$. From right to left is trivial. For the left to right direction, if $\mathbb{F}, V, w \Vdash \#\varphi \succ \#\psi$, then there are two cases:

- $[\![\#\varphi]\!]^{\mathbb{F},V,w} = 2$ and $[\![\#\psi]\!]^{\mathbb{F},V,w} < 2$;
- $[\![\#\varphi]\!]^{\mathbb{F},V,w} = 1$ and $[\![\#\psi]\!]^{\mathbb{F},V,w} = 0$.

In the first case, we have that w has two successors and $\mathbb{F}, V, w \Vdash \Box\varphi \wedge \neg\Box\psi$, in the second case, we have that $\mathbb{F}, V, w \Vdash \Diamond\varphi \wedge \neg\Diamond\psi$. $\Box$

Since Alt_1 is a subclass of Alt_2, the previous proposition also holds for Alt_1. Therefore, we can transform an ML(#)-formula into an equivalent basic modal formula. Then by the decidability of the satisfiability problem for ML in Alt_1 and Alt_2, we get the decidability of the satisfiability problem for ML(#) in Alt_1 and Alt_2.

5 Decidability of GML(#)

In this section, we consider the expansion of ML(#) with graded modal operators. For more details of graded modal logic, see [4, 5].

In the syntax of graded modal logic with counting GML(#), we have graded modalities $\Diamond_{\geq n}\varphi$ for each positive natural number n, intuitively reads "there are at least n successors satisfying φ". In what follows, we use cardinality comparison formulas in the shape $\#\varphi \gtrsim \#n$ in place of $\Diamond_{\geq n}\varphi$, for the sake of defining normal forms. We use the following abbreviations:

- $\#\varphi \succ \#n$ for $\#\varphi \gtrsim \#n+1$;
- $\#n \succ \#\varphi$ for $\neg(\#\varphi \gtrsim \#n)$;
- $\#n \gtrsim \#\varphi$ for $\neg(\#\varphi \succ \#n)$;
- $\#\varphi = \#n$ and $\#n = \#\varphi$ for $(\#\varphi \gtrsim \#n) \wedge (\#n \gtrsim \#\varphi)$.

For counting depth, we define $d(\#\varphi \gtrsim \#n) = d(\varphi) + 1$.

For the semantics of $\#\varphi \gtrsim \#n$,

$$\mathbb{M}, s \Vdash \#\varphi \gtrsim \#n \text{ iff } [\![\#\varphi]\!]^{\mathbb{M},s} \geq n.$$

When defining the normal form of GML(#), not only do we need to fix a finite number of propositional variables $p_1, \ldots, p_m$, but also we need to fix an upper bound of the number n occurring in the graded formulas $\#\varphi \gtrsim \#n$, in order to make the counterpart of the definition of k-types finite.

Now we define the notion of m, n, k-types as follows:

Definition 5 (m, n, k-type)**.** Fix m propositional variables $p_1, \ldots, p_m$ and an upper bound of the number n occurring in the graded formulas $\#\varphi \gtrsim \#n$, the k-types are defined inductively on k:

- A 0-type is a complete conjunctive clause;

- A $(k+1)$-type is a conjunction of a 0-type and a complete set of inequalities which form a linear order

$$\#T_{1,1} = \#T_{1,2} = \ldots = \#T_{1,k_1} \succ \#T_{2,1} = \#T_{2,2} = \ldots = \#T_{2,k_2} \succ$$

$$\ldots \succ \#T_{t,1} = \#T_{t,2} = \ldots = \#T_{t,k_t}$$

where $T_{1,1}, T_{1,2}, \ldots, T_{t,k_t}$ is a complete list of all formulas that are disjunctions (possibly an empty disjunction) of k-types together with all numerical constants $1, 2, \ldots, n$.

By similar arguments, we have the counterparts of Propositions 1, 2, 3, 4:

Proposition 11. *Fix m propositional variables $p_1, \ldots, p_m$ and an upper bound of the number n occurring in the graded formulas $\#\varphi \succsim \#n$, there are finitely many m, n, k-types for each k.*

Proposition 12. *Each formula φ of GML$(\#)$ with counting depth at most k is equivalent to a disjunction of m, n, k-types for some m, n, and this disjunction can be computed by an algorithm.*

Proposition 13. *For any two different m, n, k-types φ_k and ψ_k, the conjunction $\varphi_k \wedge \psi_k$ is not satisfiable.*

Proposition 14. *Fix m,n,k, suppose that $\alpha_1, \ldots, \alpha_t$ enumerate all the m, n, k-types, then $\top \leftrightarrow \alpha_1 \vee \ldots \vee \alpha_t$ is valid.*

Then we can assign inequalities to the linear order part of an m, n, k-type, where we assign the number n instead of sum of variables to the cardinality comparison formulas involving n.

Then Proposition 7 still holds, and we can apply the same algorithm as Proposition 8 to show that the satisfiability problem of GML(#) with respect to the class of all Kripke frames is decidable (notice that each formula has finitely many propositional variables and an upper bound n in cardinality comparison formulas).

References

[1] G. A. Antonelli, 2010, “Numerical abstraction via the Frege quantifier”, *Notre Dame Journal of Formal Logic*, **51(2)**: 161–179.

[2] J. van Benthem and T. Icard, 2021, “Interleaving logic and counting”, Prepublication (PP) Series.

[3] H. Herre, M. Krynicki, A. Pinus and J. Väänänen, 1991, “The Härtig quantifier: A survey”, *Journal of Symbolic Logic*, **56(4)**: 1153–1183.

[4] W. V. der Hoek, 1992, "On the semantics of graded modalities", *Journal of Applied Non-Classical Logics*, **2(1)**: 81–123.

[5] M. Ma, 2011, *Model Theory for Graded Modal Languages*, phdthesis, Tsinghua University.

[6] M. Otto, 2017, *Bounded Variable Logics and Counting: A Study in Finite Models*, Cambridge: Cambridge University Press.

[7] N. Rescher, 1962, "Plurality quantification", *Journal of Symbolic Logic*, **27(3)**: 373–374.

模态计数逻辑 ML(#) 在不同框架类下的可判定性

付小轩　　赵之光

摘　要

在本文中，我们给出模态计数逻辑 ML(#) 在不同框架类下的可满足性的判定过程。我们使用两种方法，一种是通过修改 ML(#) 相对于全部克里普克框架的可满足性的判定算法，另一种是将 ML(#) 的可判定性归约到基本模态逻辑。我们还证明了分次模态计数逻辑 GML(#) 相对于全部克里普克框架的可判定性。

付小轩　中国政法大学人文学院
xfuuva@gmail.com
赵之光　泰山学院数学与统计学院
zhaozhiguang23@gmail.com