Tableaux and Model Checking for Memory Logics

Carlos Areces[1], Diego Figueira[2], Daniel Gorín[3], and Sergio Mera[3]*

[1] INRIA Nancy Grand Est, Nancy, France,
areces@loria.fr,
[2] INRIA Saclay, ENS Cachan, LSV, France,
figueira@lsv.ens-cachan.fr,
[3] Departamento de Computación, UBA, Argentina,
{dgorin, smera}@dc.uba.ar

Abstract. Memory logics are modal logics whose semantics is specified in terms of relational models enriched with additional *data structure* to represent *memory*. The logical language is then extended with a collection of operations to access and modify the data structure. In this paper we study their satisfiability and the model checking problems.
We first give sound and complete tableaux calculi for the memory logic $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ (the basic modal language extended with the operator $\textcircled{r}$ used to memorize a state, the operator $\textcircled{e}$ used to wipe out the memory, and the operator $\textcircled{k}$ used to check if the current point of evaluation is memorized) and some of its sublanguages. As the satisfiability problem of $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ is undecidable, the tableau calculus we present is non terminating. Hence, we furthermore study a variation that ensures termination, at the expense of completeness, and we use model checking to ensure soundness. Secondly, we show that the model checking problem is PSpace-complete.

1 Memory Logics

In a number of recent papers [1–4] we have investigated a family of logics that we call *memory logics*. These logics are related to both modal logics [5, 6] and hybrid logics [7], as well as other logics that intend to add some notion of state to models [8–12].

Intuitively, memory logics are modal logics whose semantics is specified in terms of first-order relational models enriched with additional *data structure* to represent *memory*. The logical language is then extended with a collection of operations to access and modify the data structure.

Formally, let $\mathcal{M} = \langle W, (R_r)_{r \in \mathsf{Rel}}, V \rangle$ be a relational structure where W is a non empty domain; for each relation symbol r, R_r is a binary relation over W; and $V : \mathsf{Prop} \to 2^W$ is a valuation function that assigns subsets of W to propositional symbols in Prop. We can extend this structure with a set $S \subseteq W$

* S. Mera is partially supported by a grant of Fundación YPF.

which can be interpreted as a set of states that are 'known' to us, and which represent the current 'memory' of the model. Even in this simple setting we can define the following operators:

$$\begin{array}{ll}
\langle W, (R_r)_{r\in\mathsf{Rel}}, V, S\rangle, w \models \textcircled{r}\varphi & \text{iff } \langle W, (R_r)_{r\in\mathsf{Rel}}, V, S\cup\{w\}\rangle, w \models \varphi, \\
\langle W, (R_r)_{r\in\mathsf{Rel}}, V, S\rangle, w \models \textcircled{e}\varphi & \text{iff } \langle W, (R_r)_{r\in\mathsf{Rel}}, V, \emptyset\rangle, w \models \varphi, \\
\langle W, (R_r)_{r\in\mathsf{Rel}}, V, S\rangle, w \models \textcircled{k} & \text{iff } w \in S.
\end{array}$$

As it is clear from the definition above, the 'remember' operator ⓡ (a unary modality) just marks the current state as being 'known' or 'already visited', by storing it in our 'memory' S. The 'erase' operator ⓔ (also unary) wipes out the memory. These are the operators we use to update the memory. On the other hand, the zero-ary operator ⓚ (for 'known') queries S to check if the current state has already been visited. Notice that the extension of S is dynamic and it can vary during the evaluation of a formula.

Our original motivation to investigate memory logics was mainly theoretical: we were looking for a modal language that included some kind of binding mechanism (notice that ⓡ effectively binds instances of ⓚ appearing in its scope), but with a decidable satisfiability problem. The memory logic $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ (i.e., the basic modal language extended with only the ⓚ and ⓡ operators) was introduced as a weakening of the operator $\downarrow$ from the hybrid logic $\mathcal{HL}(\downarrow)$ (i.e., the basic modal language extended with nominals and the $\downarrow$ binder [7]) known to be undecidable. But, as we have shown in [2, 3] , even though the language is strictly less expressive than $\mathcal{HL}(\downarrow)$, its satisfiability problem is still undecidable.

While working with the memory operators we realised that they provide an interesting perspective on modalities and their interaction with models: they are examples of operators that *modify* the model during evaluation, and in that sense they are truly *dynamic*. They are examples of logical languages that could both *check* conditions on the model, and *modify* the model accordingly. For example, while evaluating the formula $\textcircled{r}\psi$ in a model $\mathcal{M}$, the ⓡ operator transforms $\mathcal{M}$ into a new model $\mathcal{M}'$ (by adding the current point of evaluation to the memory), and ψ is then evaluated in $\mathcal{M}'$. We could imagine other operators that modify $\mathcal{M}$ in different ways: add states, change the valuation, modify accessibility relations, etc. By investigating memory logics we want to understand the basic properties of such languages. From this perspective, memory logics would be related to other well-known logics. One example are dynamic epistemic logics [8], which are languages used to model the evolution of the knowlege of epistemic agents via *updates* to the model representing their epistemic state. Other approach comes from temporal logics with explicit global clocks (for example, the logic XCTL [9]), in which these clocks are accessed and controlled through logical operators. We believe that by studying the memory logics family we will better understand some of the basic notions and intuitions that all these languages have in common.

In this article we investigate computational aspects of two classical reasoning tasks for memory logics. In Section 2 we develop sound and complete tableau calculi. As the satisfiability problem for $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ (and hence also the one

for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$) is undecidable – and given that the calculi are sound and complete – the tableaux obtained by the application of the rules we provide might be infinite. In Section 3 we restrict these calculi so that they always produce finite tableaux, but at the expense of sacrificing completeness. For this restriction to work, we will need to perform model-checking (over an induced model) and in Section 4 we investigate the complexity of this task. Because $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ is a fragment of first-order logic, we know that the problem is in PSpace [13]. We will show that it actually is PSpace-complete.

2 Complete and Sound Tableau Calculi

In this section we will introduce a tableau calculus for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ (as we will explain below, we will actually propose calculi over two particularly interesting classes of models, and discuss also calculi for some sublogics). To make the paper self-contained, we start by introducing some notation and basic notions.

Definition 1 (Syntax). *Let* $\mathsf{Prop} = \{p_1, p_2, \dots\}$ *(the* propositional symbols*) and* $\mathsf{Rel} = \{r_1, r_2, \dots\}$ *(the* relational symbols*) be disjoint, countable infinite sets.* Forms*, the set of formulas of* $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ *over signature* $\langle \mathsf{Prop}, \mathsf{Rel} \rangle$*, is defined as:*

$$\mathsf{Forms} ::= p \mid \neg p \mid \textcircled{k} \mid \neg\textcircled{k} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle r \rangle \varphi \mid [r]\varphi \mid \textcircled{r}\varphi \mid \textcircled{e}\varphi,$$

where $p \in \mathsf{Prop}$*,* $r \in \mathsf{Rel}$ *and* $\varphi, \varphi_1, \varphi_2 \in \mathsf{Forms}$*.*

Given $\varphi \in \mathsf{Forms}$ *we will write* $\overline{\varphi}$ *for the formula obtained by computing the* negated normal form *of the negation of* φ*:*

$$\begin{array}{cccccc} \overline{p} = \neg p & \overline{\textcircled{k}} = \neg\textcircled{k} & \overline{\varphi_1 \wedge \varphi_2} = \overline{\varphi_1} \vee \overline{\varphi_2} & \overline{\langle r \rangle \varphi} = [r]\overline{\varphi} & \overline{\textcircled{r}\varphi} = \textcircled{r}\overline{\varphi} \\ \overline{\neg p} = p & \overline{\neg\textcircled{k}} = \textcircled{k} & \overline{\varphi_1 \vee \varphi_2} = \overline{\varphi_1} \wedge \overline{\varphi_2} & \overline{[r]\varphi} = \langle r \rangle \overline{\varphi} & \overline{\textcircled{e}\varphi} = \textcircled{e}\overline{\varphi} \end{array}$$

Sublogics of $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ are obtained by forbidding the use of certain operators. For example, $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ is the logic obtained by restricting to formulas in Forms not containing $\textcircled{e}$.

Definition 2 (Semantics). *Given a signature* $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Rel} \rangle$*, a* model *for* $\mathcal{S}$ *is a tuple* $\langle W, (R_r)_{r \in \mathsf{Rel}}, V, S \rangle$*, satisfying the following conditions: (i)* $W \neq \emptyset$*; (ii) each* R_r *is a binary relation on* W*; (iii)* $V : \mathsf{Prop} \to 2^W$ *is a labeling function; and (iv)* $S \subseteq W$*.*

For any model $\mathcal{M} = \langle W, (R_r)_{r \in \mathsf{Rel}}, V, S \rangle$*, we will denote with* $\mathcal{M}[w_1, \dots, w_n]$ *and* $\mathcal{M}_\emptyset$ *the models* $\langle W, (R_r)_{r \in \mathsf{Rel}}, V, S \cup \{w_1, \dots, w_n\} \rangle$ *and* $\langle W, (R_r)_{r \in \mathsf{Rel}}, V, \emptyset \rangle$ *respectively.*

Given the model $\mathcal{M} = \langle W, (R_r)_{r\in\mathsf{Rel}}, V, S\rangle$ *and* $w \in W$*, the semantics for the different operators is defined as follows:*

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \iff & w \in V(p), \quad p \in \mathsf{Prop} \\
\mathcal{M}, w \models \neg p & \iff & w \notin V(p), \quad p \in \mathsf{Prop} \\
\mathcal{M}, w \models \varphi \wedge \psi & \iff & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \varphi \vee \psi & \iff & \mathcal{M}, w \models \varphi \text{ or } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle r\rangle\varphi & \iff & \text{there is } w' \text{ such that } R_r(w, w') \text{ and } \mathcal{M}, w' \models \varphi \\
\mathcal{M}, w \models [r]\varphi & \iff & \text{for all } w' \text{ such that } R_r(w, w'),\ \mathcal{M}, w' \models \varphi \\
\mathcal{M}, w \models \textcircled{r}\varphi & \iff & \mathcal{M}[w], w \models \varphi \\
\mathcal{M}, w \models \textcircled{e}\varphi & \iff & \mathcal{M}_\emptyset, w \models \varphi \\
\mathcal{M}, w \models \textcircled{k} & \iff & w \in S \\
\mathcal{M}, w \models \neg\textcircled{k} & \iff & w \notin S.
\end{array}
$$

Definition 3 (Satisfiability and Validity). *Let* $\mathcal{C}$ *be a class of models. We say that* φ *is* satisfiable in $\mathcal{C}$ *if there is a model* $\mathcal{M} \in \mathcal{C}$ *and a state* w *in the domain of* $\mathcal{M}$ *such that* $\mathcal{M}, w \models \varphi$*. We say that* φ *is* valid in $\mathcal{C}$ *if* $\overline{\varphi}$ *is not satisfiable in* $\mathcal{C}$*.*

We will be mainly interested in using tableaux to characterize the set of valid formulas over $\mathcal{C}_{\text{all}}$, the class of all models. But observe that to express several structural properties of interest, it is natural to start with a model with no previously remembered states.

For example, $\langle W, (R_r)_{r\in\mathsf{Rel}}, \emptyset\rangle, w \models \textcircled{r}\langle r\rangle\textcircled{k}$ if and only if $R(w, w)$. That is, satisfiability of $\textcircled{r}\langle r\rangle\textcircled{k}$ at w characterizes reflexivity of w whenever the initial memory is empty. When the $\textcircled{e}$ operator is in the language, we can actually use the formula $\textcircled{e}\textcircled{r}\langle r\rangle\textcircled{k}$ instead, which ensures that S is empty before evaluating $\textcircled{r}\langle r\rangle\textcircled{k}$. That is, if $\mathcal{C}_\emptyset$ is the class $\{\mathcal{M} \mid \mathcal{M} = \langle W, (R_r)_{r\in\mathsf{Rel}}, V, \emptyset\rangle\}$ of models with an empty memory, then φ is valid in $\mathcal{C}_\emptyset$ iff $\textcircled{e}\varphi$ is valid in $\mathcal{C}_{\text{all}}$. Or in other words, we can use $\textcircled{e}$ to 'internalize' the class $\mathcal{C}_\emptyset$.

Because we will discuss not only the full language $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$, but also some of its sublanguages, we'll set up the tableau calculi so that they can be used for satisfiability for both $\mathcal{C}_{\text{all}}$ and $\mathcal{C}_\emptyset$.

In Figure 1 we present the rules for a prefixed tableau calculus for the logic $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$. Prefixed tableaux for hybrid logics were investigated by Blackburn and Bolander in [14]. The general approach and, in particular, the termination argument used in Section 3 are inspired by this paper.

A tableau in the calculus presented in Figure 1 is simply a wellfounded, finitely branching tree in which edges represent applications of tableau rules in the usual way and each node is labeled by an accessibility, equality or prefixed formula.

Definition 4 (Prefixed, accessibility and equality formulas). *Let* $W = \{w_1, w_2, \ldots\}$ *be an infinite, enumerable set of labels. Then* $\langle w, A\rangle^C{:}\varphi$ *is a* prefixed formula*, where* $\varphi \in \mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$*,* $C \in \{\mathcal{C}_{\text{all}}, \mathcal{C}_\emptyset\}$*,* $w \in W$ *and* A *is a finite subset of* W*.* $R_r(w, w')$ *is an* accessibility formula *for* $r \in \mathsf{Rel}$*, and* $w, w' \in W$*.* $w \approx w'$ *is an* equality formula *for* $w, w' \in W$*.*

$$(\wedge)\ \frac{\langle w,A\rangle^C{:}\varphi\wedge\psi}{\begin{array}{c}\langle w,A\rangle^C{:}\varphi\\ \langle w,A\rangle^C{:}\psi\end{array}} \qquad (\vee)\ \frac{\langle w,A\rangle^C{:}\varphi\vee\psi}{\langle w,A\rangle^C{:}\varphi \mid \langle w,A\rangle^C{:}\psi}$$

$$(\langle r\rangle)\ \frac{\langle w,A\rangle^C{:}\langle r\rangle\varphi}{\begin{array}{c}R_r(w,w')\\ \langle w',A\rangle^C{:}\varphi\end{array}}\ \dagger \qquad ([r])\ \frac{\begin{array}{c}\langle w,A\rangle^C{:}[r]\varphi\\ R_r(w,w')\end{array}}{\langle w',A\rangle^C{:}\varphi}$$

$$(\neg\textcircled{k})\ \frac{\langle w,A\rangle^C{:}\neg\textcircled{k}}{\langle w,\emptyset\rangle^C{:}\neg\textcircled{k}} \qquad (\textcircled{k})\ \frac{\langle w,\{v_1,\dots v_k\}\rangle^C{:}\textcircled{k}}{w\approx v_1 \mid \cdots \mid w\approx v_k \mid \langle w,\emptyset\rangle^C{:}\textcircled{k}}$$

$$(\textcircled{e})\ \frac{\langle w,A\rangle^C{:}\textcircled{e}\varphi}{\langle w,\emptyset\rangle^{C_\emptyset}{:}\varphi} \qquad (\textcircled{r})\ \frac{\langle w,A\rangle^C{:}\textcircled{r}\varphi}{\langle w,A\cup\{w\}\rangle^C{:}\varphi}$$

$$\text{(repl)}\ \frac{\begin{array}{c}\langle w,A\rangle^C{:}\varphi\\ w\approx^* w'\end{array}}{\langle w',A[w\mapsto w']\rangle^C{:}\varphi}\ \ddagger$$

Clash Rules:

$$(\bot_p)\ \frac{\begin{array}{c}\langle w,A\rangle^{C_1}{:}p\\ \langle w,B\rangle^{C_2}{:}\neg p\end{array}}{\bot} \qquad (\bot_{\textcircled{k}})\ \frac{\begin{array}{c}\langle w,\emptyset\rangle^C{:}\textcircled{k}\\ \langle w,\emptyset\rangle^C{:}\neg\textcircled{k}\end{array}}{\bot}$$

$$(\bot_{\neg\textcircled{k}})\ \frac{\langle w,\{w\}\cup A\rangle^C{:}\neg\textcircled{k}}{\bot} \qquad (\bot_\emptyset)\ \frac{\langle w,\emptyset\rangle^{\mathcal{C}_\emptyset}{:}\textcircled{k}}{\bot}$$

Key:

† w' is fresh.

‡ $a\approx^* b$ iff (a,b) occurs in the reflexive, symmetric and transitive closure of the relation $\{(w,w') \mid w\approx w' \text{appears in the current branch}\}$. $A[w\mapsto w'] = A$ if $w\notin A$, and $(A-\{w\})\cup\{w'\}$ otherwise.

- C, C_1, C_2 are either $\mathcal{C}_{\text{all}}$ or $\mathcal{C}_\emptyset$.
- A, B are arbitrary finite set of labels.

Fig. 1. Tableau rules

Intuitively, in the prefix $\langle w, A\rangle^C$, w is the label of the state where the formula holds, C is the class of models we are working with ($\mathcal{C}_{\text{all}}$ or $\mathcal{C}_\emptyset$), and A is a set of states that were explicitly remembered by evaluating a $\textcircled{\texttt{r}}$ operator in the current branch. Since every prefixed formula is derived in finitely many steps, A will always be a finite set. In the rest of this article we will refer to a tableau that uses the rules presented in Figure 1 as a tableau for $\mathcal{ML}(\textcircled{\texttt{k}}, \textcircled{\texttt{r}}, \textcircled{\texttt{e}})$. The intended interpretation of $R_r(w, w')$ is that the state denoted by w' is accessible from the state denoted by w by the interpretation of relation symbol r. Finally, the intended interpretation of an equality formula $w \approx w'$ is that w and w' label the same state in a given branch.

We will use the term *formula* to denote either a formula of $\mathcal{ML}(\textcircled{\texttt{k}}, \textcircled{\texttt{r}}, \textcircled{\texttt{e}})$, a prefixed formula, an accessibility formula, or an equality formula.

The rules are presented in the standard format: each rule has a name on the left and is divided in an upper (the antecedent) and lower (the consequent) part. Whenever there are formulas in a branch that match the antecedent, the rule can be applied following the constraints specified for each rule. If the rule is applied, the formulas of the consequent are added to the same branch, except in the case of ($\vee$) and ($\textcircled{\texttt{k}}$), where several different branches are created.

The rules ($\langle r\rangle$), ($\neg\textcircled{\texttt{k}}$), ($\textcircled{\texttt{k}}$), ($\textcircled{\texttt{e}}$), ($\textcircled{\texttt{r}}$) and (repl) are called "prefix generating rules", since if a prefix is new to a branch, it must be introduced by one of these rules. We impose two general constraints on the construction of a tableau:

- A prefix generating rule is never applied twice to a formula on a given branch.
- A formula is never added to a tableau branch where it already occurs.

A *saturated tableau* is a tableau in which no more rules can be applied that satisfy the constraints. A *saturated branch* is a branch of a saturated tableau. A branch of a tableau is called *closed* if it contains $\bot$. Otherwise the branch is called *open*. A *closed tableau* is one in which all branches are closed, and an *open tableau* is one in which at least one branch is open.

($\wedge$), ($\vee$), ($\langle r\rangle$) and ($[r]$) are classical rules of the basic modal logic tableau calculus. The remaining ones are particular to memory logics. Rule ($\neg\textcircled{\texttt{k}}$) specifies that at a label where A denotes the set of states that were explicitly remembered, if the state w is not in the memory then $w \notin A$ and (in particular) w still is not memorized at the label with $A = \emptyset$. Rule ($\textcircled{\texttt{k}}$) specifies that if w is in the memory, then either it is one of the explicitly remembered states, or it is in the initial memory, in which case $\textcircled{\texttt{k}}$ holds even with no explicitly remembered states. Notice that the last branch of the application of this rule can be immediately closed in the case where $C = C_\emptyset$, due to the rule ($\bot_\emptyset$). Rule ($\textcircled{\texttt{e}}$) wipes out the explicitly remembered states and evaluates the satisfiability of the formula in a model with no initial memory. Observe that the presence of the $\textcircled{\texttt{e}}$ modality may force the calculus to switch from the evaluation over C_{all} to that over $C_\emptyset$.

We will also consider variations and subsystems of the calculus of Figure 1 where only a subset of the rules are allowed, or additional constraints on the rules are imposed (for example, to ensure termination). In such subsystems, a tableau is of course simply a tableau in which only the rules in the subset can be applied, considering the additional constraints.

We call a tableau calculus $\mathcal{T}$ *sound* for a language $\mathcal{L}$ respect to a class of models $\mathcal{C}$ if whenever $\varphi \in \mathcal{L}$ is $\mathcal{C}$-satisfiable, then every saturated tableau T with root φ has an open branch. We say that it is *complete* if whenever $\varphi \in \mathcal{L}$ is not $\mathcal{C}$-satisfiable, then every saturated tableau T with root φ is closed.

Soundness of the calculus of Figure 1 follows from a simple inspection of the rules. We devote the rest of this section to prove completeness. As usual, we will show that given an open and saturated branch Γ, we can define a model $\mathcal{M}^\Gamma$ that satisfies all the formulas that occur in the branch. To define the domain of $\mathcal{M}^\Gamma$ we first need the following definition.

Definition 5 (Eq_Γ)**.** *Let Γ be an open and saturated branch of a tableau for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$. Eq_Γ is the smallest equivalence relation extending $\{(w, w') \mid (w \approx w') \in \Gamma\}$.*

Definition 6 ($\mathcal{M}^\Gamma$)**.** *Let Γ be an open and saturated branch of a tableau for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$. Define the induced model $\mathcal{M}^\Gamma = \langle W_\Gamma, (R_{r\Gamma})_{r \in \mathsf{Rel}}, V_\Gamma, S_\Gamma \rangle$ as:*

$$\begin{array}{ll} W_\Gamma & = \{w \mid w \text{ occurs in } \Gamma\}/\mathrm{Eq}_\Gamma \\ R_{r\Gamma} & = \{([w], [w']) \mid R_r(w, w') \in \Gamma\} \\ V_\Gamma(p) & = \{[w] \mid \langle w, A\rangle^C{:}p \in \Gamma, \text{ for any } A \text{ and } C\} \\ S_\Gamma & = \{[w] \mid \langle w, \emptyset\rangle^{\mathcal{C}_{all}}{:}\textcircled{k} \in \Gamma\}, \end{array}$$

where $[w]$ is the equivalence class of w in Eq_Γ.

Lemma 1. *Let $\mathcal{M}^\Gamma = \langle W_\Gamma, (R_{r\Gamma})_{r \in \mathsf{Rel}}, V_\Gamma, S_\Gamma \rangle$ be the induced model for Γ, where Γ is an open and saturated branch of a tableau for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$.*

1. *$\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_{all}}{:}\varphi \in \Gamma$ implies $\mathcal{M}^\Gamma[[v_1], \ldots, [v_k]], [w] \models \varphi$.*
2. *$\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_\emptyset}{:}\varphi \in \Gamma$ implies $\mathcal{M}^\Gamma_\emptyset[[v_1], \ldots, [v_k]], [w] \models \varphi$.*

Proof. We proceed by induction on φ.

Case $\varphi := p$**.** If $\langle w, \{v_1, \ldots, v_k\}\rangle^{\mathcal{C}_{\mathrm{all}}}{:}p \in \Gamma$, then $[w] \in V_\Gamma(p)$ and, therefore, $\mathcal{M}^\Gamma[[v_1], \ldots, [v_k]], [w], \models p$. The case for $\mathcal{C}_\emptyset$ is analogous.

Case $\varphi := \neg p$**.** Suppose $\langle w, \{v_1, \ldots, v_k\}\rangle^C{:}\neg p \in \Gamma$. If $\mathcal{M}^\Gamma[[v_1], \ldots, [v_k]], [w] \models p$, it means that $[w] \in V_\Gamma(p)$ and hence $\langle w, A\rangle^C{:}p \in \Gamma$ (for some A and C), but in that case rule $(\bot_p)$ applies and the branch would be closed. Again, the case for $\mathcal{C}_\emptyset$ is analogous.

Case $\varphi := \textcircled{k}$**.** We consider all the different possibilities:

1. If $\langle w, \emptyset\rangle^{\mathcal{C}_{\mathrm{all}}}{:}\textcircled{k} \in \Gamma$, then $[w] \in S_\Gamma$ and, therefore, $\mathcal{M}^\Gamma, [w] \models \textcircled{k}$.
2. If $\langle w, \emptyset\rangle^{\mathcal{C}_\emptyset}{:}\textcircled{k} \in \Gamma$, then, by the $(\bot_\emptyset)$ rule, $\bot \in \Gamma$ which would contradict the hypothesis that Γ is an open branch.
3. If $\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_{\mathrm{all}}}{:}\textcircled{k} \in \Gamma$, with $k > 0$ then some consequent of the $(\textcircled{k})$ rule must occur in Γ too. If $\langle w, \emptyset\rangle^{\mathcal{C}_{\mathrm{all}}}{:}\textcircled{k} \in \Gamma$ then we are done. So let us assume that, on the contrary, $w \approx v_i \in \Gamma$ for some $i \in \{1, \ldots k\}$. This implies that $[w] = [v_i]$, but since $v_i \in \{v_1, \ldots v_k\}$, we conclude that $\mathcal{M}^\Gamma[[v_1], \ldots, [v_k]], [w] \models \textcircled{k}$.
4. The case when $\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_\emptyset}{:}\textcircled{k} \in \Gamma$, with $k > 0$ is analogous.

Case $\varphi := \neg\textcircled{\scriptsize k}$**.** We consider, again, all the distinct cases:

1. Let $\langle w, \emptyset\rangle^{\mathcal{C}_{\text{all}}}{:}\neg\textcircled{\scriptsize k} \in \Gamma$ and let us assume, for the sake of contradiction, that $\mathcal{M}^{\Gamma}, [w] \models \textcircled{\scriptsize k}$. This means that $[w] \in S_{\Gamma}$ and, therefore, $\langle w', \emptyset\rangle^{\mathcal{C}_{\text{all}}}{:}\textcircled{\scriptsize k} \in \Gamma$, where $[w] = [w']$. Since Γ is saturated, by the (repl) rule we know that $\langle w, \emptyset\rangle^{\mathcal{C}_{\text{all}}}{:}\textcircled{\scriptsize k} \in \Gamma$. But then rule $(\bot_{\textcircled{\scriptsize k}})$ applies and $\bot \in \Gamma$ which makes Γ a closed branch.
2. Suppose $\langle w, \emptyset\rangle^{\mathcal{C}_{\emptyset}}{:}\neg\textcircled{\scriptsize k} \in \Gamma$. It is always the case that $\mathcal{M}^{\Gamma}_{\emptyset}, [w] \models \neg\textcircled{\scriptsize k}$.
3. Let $\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_{\text{all}}}{:}\neg\textcircled{\scriptsize k} \in \Gamma$, with $k > 0$, and suppose, for the sake of contradiction, that $\mathcal{M}^{\Gamma}[[v_1], \ldots, [v_k]], [w] \models \textcircled{\scriptsize k}$. This opens two possibilities. First, it could be the case that $[w] \in S_{\Gamma}$, but that would mean that $\langle w, \emptyset\rangle^{\mathcal{C}_{\text{all}}}{:}\textcircled{\scriptsize k} \in \Gamma$ and, because of the $(\neg\textcircled{\scriptsize k})$ rule, $\langle w, \emptyset\rangle^{\mathcal{C}_{\text{all}}}{:}\neg\textcircled{\scriptsize k} \in \Gamma$ and therefore we would have a clash by the $(\bot_{\textcircled{\scriptsize k}})$ rule.

   Alternatively, it could be the case that $[w] = [v_i]$ for some $i \in \{1, \ldots k\}$. Since Γ is saturated, we conclude $\langle v_i, \{v_1, \ldots v_k\}[w \mapsto v_i]\rangle^{\mathcal{C}_{\text{all}}}{:}\neg\textcircled{\scriptsize k} \in \Gamma$ using the (repl) rule. But observe that $v_i \in \{v_1, \ldots v_k\}[w \mapsto v_i]$ from which rule $(\bot_{\neg\textcircled{\scriptsize k}})$ applies and leads to a contradiction.
4. If $\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_{\emptyset}}{:}\neg\textcircled{\scriptsize k} \in \Gamma$, with $k > 0$, we can simply use the argument for the case $[w] = [v_i]$ just above.

Case $\varphi := \textcircled{\scriptsize r}\psi$**.** Suppose $\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_{\text{all}}}{:}\textcircled{\scriptsize r}\psi \in \Gamma$. By rule $(\textcircled{\scriptsize r})$, we know $\langle w, \{v_1, \ldots v_k, w\}\rangle^{\mathcal{C}_{\text{all}}}{:}\psi \in \Gamma$. By inductive hypothesis, $\mathcal{M}^{\Gamma}[[v_1], \ldots [v_k], [w]], [w] \models \psi$, which implies $\mathcal{M}^{\Gamma}[[v_1], \ldots [v_k]], [w] \models \textcircled{\scriptsize r}\psi$. $\mathcal{C}_{\emptyset}$ is analogous.

Case $\varphi := \textcircled{\scriptsize e}\psi$**.** If $\langle w, \{v_1, \ldots v_k\}\rangle^{\mathcal{C}_{\text{all}}}{:}\textcircled{\scriptsize e}\psi \in \Gamma$ then, by rule $(\textcircled{\scriptsize e})$, $\langle w, \emptyset\rangle^{\mathcal{C}_{\emptyset}}{:}\psi \in \Gamma$ and, by inductive hypothesis, $\mathcal{M}^{\Gamma}_{\emptyset}, [w] \models \psi$. Therefore, it follows that $\mathcal{M}[[v_1], \ldots [v_k]], [w] \models \textcircled{\scriptsize e}\psi$. The case for $\mathcal{C}_{\emptyset}$ is analogous.

The remaining boolean and modal cases are dealt with in the standard way.

Theorem 1. *The tableau calculus for* $\mathcal{ML}(\textcircled{\scriptsize k}, \textcircled{\scriptsize r}, \textcircled{\scriptsize e})$ *is sound and complete for both the classes* $\mathcal{C}_{\text{all}}$ *and* $\mathcal{C}_{\emptyset}$*.*

More precisely, given $\varphi \in \mathcal{ML}(\textcircled{\scriptsize k}, \textcircled{\scriptsize r}, \textcircled{\scriptsize e})$*,* φ *is satisfiable iff any saturated tableau for* $\mathcal{ML}(\textcircled{\scriptsize k}, \textcircled{\scriptsize r}, \textcircled{\scriptsize e})$ *with root* $\langle w, \emptyset\rangle^{\mathcal{C}_{\text{all}}}{:}\varphi$ *has an open branch. An equivalent result holds for the* $\mathcal{C}_{\emptyset}$ *class, starting with a tableau with root* $\langle w, \emptyset\rangle^{\mathcal{C}_{\emptyset}}{:}\varphi$*.*

Proof. Soundness is trivial. Completeness is straightforward from Lemma 1: assume that a formula $\varphi \in \mathcal{ML}(\textcircled{\scriptsize k}, \textcircled{\scriptsize r}, \textcircled{\scriptsize e})$ is not satisfiable in the class C while there is a saturated tableau T with root $\langle w, \emptyset\rangle^{C}{:}\varphi$ and open branch Γ; $\mathcal{M}^{\Gamma}$ satisfies φ and is in the class C which contradicts the assumption.

It is also straightforward to see that if we drop the $(\textcircled{\scriptsize e})$ rule from the calculus, then we can prove soundness and completeness for formulas in $\mathcal{ML}(\textcircled{\scriptsize r}, \textcircled{\scriptsize k})$ (again with respect to both classes $\mathcal{C}_{\text{all}}$ and $\mathcal{C}_{\emptyset}$).

Theorem 2. *The tableau calculus of Figure 1 without the* $(\textcircled{\scriptsize e})$ *rule is sound and complete for* $\mathcal{ML}(\textcircled{\scriptsize r}, \textcircled{\scriptsize k})$ *for both the classes* $\mathcal{C}_{\text{all}}$ *and* $\mathcal{C}_{\emptyset}$*.*

3 Terminating Tableaux

In this section we will investigate some constraints that can be applied to the tableau rules for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ in order to ensure termination. The price we have to pay is that the resulting calculus is not complete any more. More precisely, it will be the case that some formula φ has a tableau with an open saturated branch Γ whose induced model $\mathcal{M}^\Gamma$ is not a model for φ. This means that we cannot claim satisfiability of the root formula every time we obtain a saturated open tableau. In these cases, we will use a model checking algorithm to verify whether $\mathcal{M}^\Gamma$ is effectively a model for φ. If the model checking succeeds we can then indeed answer SAT, and we will answer NOT-KNOWN otherwise.

We begin this section defining the restricted tableau rules, and proving a termination result. After this we will formalize the connection with model checking. In what follows, when a prefixed formula $\sigma{:}\varphi$ occurs in a tableau branch Γ we will write $\sigma{:}\varphi \in \Gamma$, and say that φ is true at σ on Γ or that σ makes φ true on Γ. Also, given a prefix $\sigma = \langle w, A\rangle^C$ we will define $\mathit{Label}(\sigma) = w$ and $\mathit{Set}(\sigma) = A$.

We will start by showing that by eliminating the (repl) rule one obtains a terminating calculus.

Definition 7. *Given a tableau branch Γ and a prefix σ, the set of true formulas at σ on Γ, written $T_\Gamma(\sigma)$, is defined as $T_\Gamma(\sigma) = \{\varphi \mid \sigma{:}\varphi \in \Gamma\}$.*

Notice that accessibility and equality formulas are not included in $T_\Gamma(\sigma)$.

Lemma 2 (Subformula Property). *Let T be a tableau with the prefixed formula $\sigma_0{:}\varphi_0$ as root. For any prefixed formula $\sigma{:}\varphi$ occurring on T, φ is a subformula of φ_0.*

Proof. This is easily seen by going through each of the tableau rules.

Lemma 3. *Let Γ be a branch of a tableau, and let σ be any prefix occurring on Γ. The set $T_\Gamma(\sigma)$ is finite.*

Proof. Let $\sigma_0{:}\varphi_0$ denote the first formula on Γ (i.e., the root of the tableau). From Lemma 2, we know that $T_\Gamma(\sigma) \subseteq \{\varphi \mid \varphi \text{ is a subformula of } \varphi_0\}$, and hence $T_\Gamma(\sigma)$ is finite.

Definition 8. *Let T be a tableau. If a prefixed formula $\tau{:}\psi$ of T has been introduced by applying one of the prefix generating rules except* (repl) *to a premise $\sigma{:}\varphi$ of T then we say that $\tau{:}\psi$ is generated by $\sigma{:}\varphi$, and we write $\sigma : \varphi \prec \tau{:}\psi$.*

Now we define a measure for the complexity of a prefixed formula:

Definition 9. *Let T be a tableau, $\sigma{:}\varphi$ be a prefixed formula occurring on T and let $|\varphi|$ denote the length of the φ. We define*

$$m(\sigma{:}\varphi) = 2|\varphi| + |\mathit{Set}(\sigma)|,$$

Lemma 4 (Decreasing length). *Let T be a tableau with no application of the* (repl) *rule. If $\sigma{:}\psi \prec \tau{:}\varphi$ then $m(\sigma{:}\psi) > m(\tau{:}\varphi)$.*

Proof. Assume $\sigma{:}\psi \prec \tau{:}\varphi$. We need to prove $m(\sigma{:}\psi) > 2|\varphi| + |Set(\tau)|$. $\tau{:}\varphi$ must have been introduced by an application of either $(\langle r\rangle)$, $([r])$, $(\textcircled{k})$, $(\neg\textcircled{k})$, $(\textcircled{e})$, $(\textcircled{r})$, $(\wedge)$ or $(\vee)$.

In the case of $(\langle r\rangle)$, $\tau{:}\varphi$ must be introduced by applying the $(\langle r\rangle)$ rule to a premise of the form $\sigma{:}\langle r\rangle\varphi$. In the case of $([r])$, $\tau{:}\varphi$ must be introduced by applying the $([r])$ rule to a pair of premises of the form $\sigma{:}[r]\varphi$, $R_r(Label(\sigma), Label(\tau))$. In both cases we see that $\tau{:}\varphi$ is introduced by applying a rule to a formula $\sigma{:}\psi$ where $|\psi| > |\varphi|$ and where $|Set(\tau)| = |Set(\sigma)|$. Thus we get $m(\sigma{:}\psi) = |Set(\sigma)| + 2|\psi| > |Set(\tau)| + 2|\varphi|$.

If $\tau{:}\varphi$ is introduced by $(\textcircled{k})$ or $(\neg\textcircled{k})$ from $\sigma{:}\psi$, it is immediate that $\varphi = \psi$, $|Set(\tau)| = 0$ and also that $|Set(\sigma)| > 0$, because otherwise the application would generate a prefixed formula already in the branch. Thus, $m(\sigma{:}\psi) = |Set(\sigma)| + 2|\psi| > 0 + 2|\psi| = |Set(\tau)| + 2|\varphi|$. The case of $(\textcircled{e})$ is clear as the length of the set does not increase, and the length of the formula decreases. In the cases of $(\vee)$ and $(\wedge)$ the length of the formula is decreased while the set is preserved.

Finally, if $\tau{:}\varphi$ is introduced by the $(\textcircled{r})$ rule from $\sigma{:}\psi$, we see that while the set may be increased by one, the length of the formula is always decremented. Then we have $m(\sigma{:}\psi) = |Set(\sigma)| + 2|\psi| > (|Set(\sigma)| + 1) + 2(|\psi| - 1) = |Set(\tau)| + 2|\varphi|$.

Lemma 5 (Finite branching). *Let Γ be a branch of a tableau. For any $\sigma{:}\varphi \in \Gamma$ there is only a finite number of prefixed formulas $\tau{:}\psi \in \Gamma$ such that $\sigma{:}\varphi \prec \tau{:}\psi$.*

Proof. For any given prefix σ the set $T_\Gamma(\sigma)$ is finite (Lemma 3), and for each formula $\varphi \in T_\Gamma(\sigma)$ at most one new prefix has been generated from σ (by applying a prefix generating rule to $\sigma{:}\varphi$). Thus $\prec$ is finitely branching.

Theorem 3. *Fix a natural number $n \geq 0$. Any tableau for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ in which the rule* (repl) *is applied at most n times per branch is finite.*

Proof. We show that any branch Γ of the tableau is finite.

Notice first that $\sigma_0{:}\varphi_0$ has no $\prec$-predecessors, and that at most $k \leq n$ other prefixed formulas of the tableau share the property of not having $\prec$-predecessors. Intuitively, each of these k formulas were introduced in Γ by the (repl) rule and hence cannot have been derived by $\prec$. We shall refer to these $k + 1$ formulas as 'generating formulas'.

It is easy to see that each generating formula induces a connected component in the graph of $\prec$. Then, every $\sigma{:}\varphi \in \Gamma$ belongs to (at least) one of these $k + 1$ connected components. As the function m decreases monotonously along any path of each of the connected components (Lemma 4), all paths of the component are finite.

By construction, there is a path between a generating formula and every node of its connected component. Then the graph is weakly connected and every path is finite. By König's Lemma the connected component is either finite or has infinite branching. As we know by Lemma 5 that it has finite branching, Γ must be finite.

Since we limit the number of applications of (repl) to n, we may have a saturated open tableau for φ whose induced model $\mathcal{M}^\Gamma$ is not a model for φ

(recall that that we are taking into account the constraint on the number of applications of (repl) when talking about *saturation*). This implies that it is no longer safe to answer SAT in these cases. But we can try to identify, given a formula φ, whether $\mathcal{M}^\Gamma$ is indeed a model for φ. The algorithm we propose is outlined as follows:

1. Given a formula φ and a parameter $n \geq 0$, build T, a saturated tableau for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ with root $\langle w, \emptyset \rangle^{\mathcal{C}_{\text{all}}}{:}\varphi$ using at most n applications of the (repl) rule per branch.
2. If T is closed answer UNSAT.
3. Else, if T has an open branch Γ, compute the induced model $\mathcal{M}^\Gamma$.
4. If $\mathcal{M}^\Gamma, [w] \models \varphi$ then answer SAT.
5. Else, answer NOT-KNOWN.

Correctness of this algorithm is straightforward. Moreover, as we will show in Section 4, $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ is a fragment of $\mathcal{HL}(\downarrow)$, and therefore we can use a model checking algorithm for $\mathcal{HL}(\downarrow)$ to perform the step 4 in polynomial space.

Note that in the case the algorithm returns NOT-KNOWN, we can try refining the result running the algorithm again with a bigger n reusing the previously computed tableau, as the resulting tableau will be an *extension* to the previous one. This method allows us to approximate increasingly to a solution to the satisfiability problem of $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ in a controlled way.

4 Model Checking

In this section we will show that the complexity of the model checking problem for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ is PSpace-complete (actually the result already holds for $\mathcal{ML}(\textcircled{k}, \textcircled{r})$). To prove the lower bound we reduce the PSpace-complete satisfiability problem for Quantified Boolean Formulas (QBF) [15] to the model checking problem of $\mathcal{ML}(\textcircled{k}, \textcircled{r})$. To prove the upper bound, we show an equivalent preserving translation from formulas of $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ to formulas of the hybrid logic $\mathcal{HL}(\downarrow)$ [7, 16].

This high complexity contrasts with the linear complexity (in both formula and model size) of model checking for the basic modal logic [17], and can be seen as a strengthening of the result of PSpace-hardness of $\mathcal{HL}(\downarrow)$ shown in [16] (in the sense that $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ is a logic with strictly weaker expressive power than $\mathcal{HL}(\downarrow)$, but whose model checking problem is already PSpace-hard).

We start by giving a lower bound for $\mathcal{ML}(\textcircled{k}, \textcircled{r})$. Since $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ is a sublanguage of $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$, the result also holds for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$.

Theorem 4. *Model checking for $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ is PSpace-hard.*

Proof. We prove it by giving a polynomial-time reduction of QBF-SAT, known to be PSpace-complete [15], to the model checking problem of $\mathcal{ML}(\textcircled{k}, \textcircled{r})$.

Let α be a QBF formula with propositional variables $\{x_1, \ldots x_k\}$. Without loss of generality, we assume that α has no free-variables and no variable

is quantified twice. One can build in polynomial time the relational structure $\mathcal{M}^k = \langle W, \{R_r\}, V, S\rangle$, over a signature with one relation symbol and propositions $\{p_\top, p_{x_1}, \ldots p_{x_k}\}$, where

$$\begin{array}{ll} V(p_{x_i}) = \{w_{x_i}\} \text{ for all } i \in [1..k] & S = \emptyset \\ V(p_\top) = \{w^\top_{x_1}, w^\top_{x_2}, \ldots w^\top_{x_k}\} & W = \{w\} \cup \{w_{x_i}, w^\top_{x_i}, w^\bot_{x_i} \mid i \in [1..k]\} \\ R_r = \{(w, w_{x_i}), (w_{x_i}, w^\top_{x_i}), (w^\top_{x_i}, w), (w_{x_i}, w^\bot_{x_i}), (w^\bot_{x_i}, w) \mid i \in [1..k]\}. & \end{array}$$

Figure 2 depicts $\mathcal{M}^k$ for $k = 3$. Let Tr be the following linear-time translation:

$$\begin{array}{ll} \mathsf{Tr}(x_i) := \langle r\rangle(p_{x_i} \wedge \langle r\rangle(p_\top \wedge \textcircled{k})) & \mathsf{Tr}(\exists x_i.\alpha) := \langle r\rangle(p_{x_i} \wedge \langle r\rangle\textcircled{r}\langle r\rangle\mathsf{Tr}(\alpha)) \\ \mathsf{Tr}(\neg\alpha) := \overline{\mathsf{Tr}(\alpha)} & \mathsf{Tr}(\alpha \wedge \beta) := \mathsf{Tr}(\alpha) \wedge \mathsf{Tr}(\beta) \end{array}$$

It only remains to see that α is satisfiable iff $\mathcal{M}^k, w \models \mathsf{Tr}(\alpha)$ holds, but this is relatively straightforward. Let us write $v \models_{\text{qbf}} \alpha$ if valuation $v : \{x_1, \ldots x_k\} \to 2$ satisfies α. For a *memory* $S \subseteq W$, define $v_S : \{x_1, \ldots x_k\}$ as "$v_S(x_i) = 1$ iff $w^\top_{x_i} \in S$". Let β be any subformula of α; we will now show by induction on β that $\langle W, V, R_r, S\rangle, w \models \mathsf{Tr}(\beta)$ iff $v_S \models_{\text{qbf}} \beta$ whenever S satisfies i) if x_i is free in β, then $w^\top_{x_i} \in S$ or $w^\bot_{x_i} \in S$ but not both, and ii) if x_i is not free in β then $w^\top_{x_i} \notin S$ and $w^\bot_{x_i} \notin S$. Observe that from here it will follow that $\mathcal{M}^k, w \models \mathsf{Tr}(\alpha)$ iff $v \models_{\text{qbf}} \alpha$ for any v (since α has no free variables) iff α is satisfiable.

For the base case, $v_S \models_{\text{qbf}} x_i$ iff $w^\top_{x_i} \in S$ which implies (from the definition of $\mathcal{M}^k$) $\langle W, V, R_r, S\rangle, w \models \mathsf{Tr}(x_i)$. For the other direction, suppose now that $\langle W, V, R_r, S\rangle, w \not\models \mathsf{Tr}(x_i)$. This means that $\langle W, V, R_r, S\rangle, w \models [r](\neg p_{x_i} \vee [r](\neg p_\top \vee \neg\textcircled{k}))$ which implies $\langle W, V, R_r, S\rangle, w_{x_i} \models [r](\neg p_\top \vee \neg\textcircled{k})$ which implies $\langle W, V, R_r, S\rangle, w^\top_{x_i} \models \neg\textcircled{k}$ and, thus, $w^\top_{x_i} \notin S$. Therefore we have $v_S \not\models_{\text{qbf}} x_i$.

Consider now the case $\beta = \exists x_i.\gamma$. Since α has no rebound variables we know $w^\top_{x_i} \notin S$ and $w^\bot_{x_i} \notin S$. We have $v_S \models_{\text{qbf}} \beta$ iff $v_S[x_i \mapsto 0] \models_{\text{qbf}} \gamma$ or $v_S[x_i \mapsto 1] \models_{\text{qbf}} \gamma$ iff $v_{S\cup\{w^\bot_{x_i}\}} \models_{\text{qbf}} \gamma$ or $v_{S\cup\{w^\top_{x_i}\}} \models_{\text{qbf}} \gamma$ iff, by inductive hypothesis, $\langle W, V, R_r, S \cup \{w^\bot_{x_i}\}\rangle \models \mathsf{Tr}(\gamma)$ or $\langle W, V, R_r, S \cup \{w^\top_{x_i}\}\rangle \models \mathsf{Tr}(\gamma)$ iff $\langle W, V, R_r, S\rangle, w^\bot_{x_i} \models \textcircled{r}\langle r\rangle\mathsf{Tr}(\gamma)$ or $\langle W, V, R_r, S\rangle, w^\top_{x_i} \models \textcircled{r}\langle r\rangle\mathsf{Tr}(\gamma)$ iff $\langle W, V, R_r, S\rangle, w \models \langle r\rangle(p_{x_i} \wedge \langle r\rangle\textcircled{r}\langle r\rangle\mathsf{Tr}(\gamma))$ iff $\langle W, V, R_r, S\rangle, w \models \mathsf{Tr}(\exists x_i.\gamma)$.

The boolean cases follow directly from the inductive hypothesis.

To see that $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ is in PSpace it is enough to show that any $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ formula can be translated to an equivalent formula of $\mathcal{H}(\downarrow)$, whose model checking problem is known to be PSpace-complete [16]. Recall that the language of $\mathcal{HL}(\downarrow)$ is the language of the basic modal logic extended with nominals and the $\downarrow$ binder (see [7] for details). Formulas of $\mathcal{HL}(\downarrow)$ are also interpreted over relational structures, but we need additionally an assignment function to interpret nominals and $\downarrow$. More formally, to evaluate a formula of $\mathcal{HL}(\downarrow)$, we need a relational structure $\mathcal{M} = \langle W, (R_r)_{r\in\mathsf{Rel}}, V\rangle$ (where W is a non empty set, each R_r is a binary relation over W, and V is a valuation function), and an assignment function g such that for any nominal i, $g(i) \in W$. Given a relational structure $\mathcal{M} = \langle W, (R_r)_{r\in\mathsf{Rel}}, V\rangle$ and an assignment g, the semantic conditions

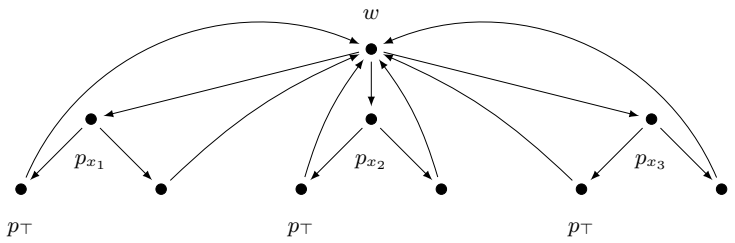

Fig. 2. $\mathcal{M}^k$ for $k = 3$.

for the $\downarrow$ operator and the nominals is defined as

$$\begin{aligned}\mathcal{M}, g, w &\models i \quad \text{iff } g(i) = w\\ \mathcal{M}, g, w &\models \downarrow i.\varphi \text{ iff } \mathcal{M}, g', w \models \varphi \text{ where } g' \text{ is identical to } g\\ &\qquad\qquad \text{except perhaps in that } g'(i) = w.\end{aligned}$$

The semantics for the other operators is the same as for the basic modal logic. Formulas in which any nominal i appears in the scope of a binder $\downarrow i$ are called *sentences*.

In order to define a translation between $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ and $\mathcal{HL}(\downarrow)$ we have to find a mapping between the models of each logic. Since $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$-models may have a nonempty memory, we must introduce a shift in the signature of $\mathcal{HL}(\downarrow)$-models to encode this information. We will associate every $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$-model $\mathcal{M} = \langle W, (R_r)_{r\in\mathsf{Rel}}, V, S\rangle$ over the signature $\langle \mathsf{Prop}, \mathsf{Rel}\rangle$ with the $\mathcal{HL}(\downarrow)$-model $\mathcal{M}' = \langle W, (R_r)_{r\in\mathsf{Rel}}, V'\rangle$ over the signature $\langle \mathsf{Prop} \cup \{known\}, \mathsf{Rel}, \mathsf{Nom}\rangle$, where V' is identical to V over Prop, and $V'(known) = S$.

Theorem 5. *Model checking for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ is PSpace-complete.*

Proof. We define the translation Tr, taking formulas of $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ over the signature $\langle \mathsf{Prop}, \mathsf{Rel}\rangle$ to $\mathcal{HL}(\downarrow)$ sentences over the signature $\langle \mathsf{Prop} \cup \{known\}, \mathsf{Rel}, \mathsf{Nom}\rangle$. Tr is defined for any finite set $N \subseteq \mathsf{Nom}$ and $C \in \{\mathcal{C}_{\text{all}}, \mathcal{C}_\emptyset\}$ as follows:

$$\begin{aligned}\mathsf{Tr}_{N,C}(p) &= p \quad p \in \mathsf{Prop}\\ \mathsf{Tr}_{N,C}(\neg p) &= \neg p \quad p \in \mathsf{Prop}\\ \mathsf{Tr}_{N,C}(\textcircled{k}) &= \begin{cases}(\bigvee_{i\in N} i) \vee known & \text{if } C = \mathcal{C}_{\text{all}}\\ \bigvee_{i\in N} i & \text{if } C = \mathcal{C}_\emptyset\end{cases}\\ \mathsf{Tr}_{N,C}(\neg\textcircled{k}) &= \begin{cases}(\bigwedge_{i\in N} \neg i) \wedge \neg known & \text{if } C = \mathcal{C}_{\text{all}}\\ \bigwedge_{i\in N} \neg i & \text{if } C = \mathcal{C}_\emptyset\end{cases}\\ \mathsf{Tr}_{N,C}(\varphi_1 \wedge \varphi_2) &= \mathsf{Tr}_{N,C}(\varphi_1) \wedge \mathsf{Tr}_{N,C}(\varphi_2)\\ \mathsf{Tr}_{N,C}(\varphi_1 \vee \varphi_2) &= \mathsf{Tr}_{N,C}(\varphi_1) \vee \mathsf{Tr}_{N,C}(\varphi_2)\end{aligned}$$

$$
\begin{aligned}
\mathsf{Tr}_{N,C}(\langle r\rangle\varphi) &= \langle r\rangle \mathsf{Tr}_{N,C}(\varphi)\\
\mathsf{Tr}_{N,C}([r]\varphi) &= [r] \mathsf{Tr}_{N,C}(\varphi)\\
\mathsf{Tr}_{N,C}(\textcircled{r}\varphi) &= \downarrow i.\mathsf{Tr}_{N\cup\{i\},C}(\varphi) \quad \text{where } i \notin N.\\
\mathsf{Tr}_{N,C}(\textcircled{e}\varphi) &= \mathsf{Tr}_{\emptyset,\mathcal{C}_\emptyset}(\varphi).
\end{aligned}
$$

A simple induction shows that, given a formula $\varphi \in \mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$, $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}', g, w \models \mathsf{Tr}_{\emptyset,\mathcal{C}_{\text{all}}}(\varphi)$ for any g.

5 Conclusions, Related and Further Work

The family of memory logics has been introduced to investigate, in the simplest possible set up, the idea of models with a dynamic state. From that perspective they are closely related to Dynamic Epistemic Logics (DELs) as those discussed in [8] and many others [9–12]. Compared to these domain-specific logics, the goals of memory logics are humbler, focusing on developing a suitable proof and model theory for logics whose semantics is defined using models that can evolve during the evaluation of a formula. From a purely formal point of view they are closer to hybrid logics. And the logic $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ that we investigated in this paper is closely related, but expressively weaker, than the logic $\mathcal{HL}(\downarrow)$ [7].

It was already proved in [2, 3] that the satisfiability problem of $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ was undecidable. In this paper we develop sound and complete tableau calculi for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ and $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ (Theorems 1 and 2) which, given the undecidability result, are non terminating. By restricting the application of one of the rules in the calculi we can obtain termination at the expense of completeness (Theorem 3). To ensure soundness of this calculus we need to perform model checking whenever we obtain an open branch. Theorem 4 shows that the model checking problem for $\mathcal{ML}(\textcircled{k}, \textcircled{r})$ is PSpace-complete.

To close the paper, we discuss how the tableau calculus for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e})$ could be extended to cover another interesting memory operator. Define the *forget* operator $\textcircled{f}$ as follows:

$$
\mathcal{M}, w \models \textcircled{f}\varphi \iff \langle W, (R_r)_{r\in\mathsf{Rel}}, V, S - \{w\}\rangle, w \models \varphi.
$$

The $\textcircled{f}$ operator gives us a fine control on which elements we want to eliminate from the memory of the model. Prefixes in the calculus for $\mathcal{ML}(\textcircled{k}, \textcircled{r}, \textcircled{e}, \textcircled{f})$ will have to explicitly record forgotten worlds in a separate set (it is not enough to simply eliminate them from the set of remembered labels). For example, the rules for ($\textcircled{f}$) and ($\textcircled{r}$) would be

$$
(\textcircled{f})\ \frac{\langle w, R, F\rangle^C{:}\textcircled{f}\varphi}{\langle w, R - \{w\}, F \cup \{w\}\rangle^C{:}\varphi} \qquad (\textcircled{r})\ \frac{\langle w, R, F\rangle^C{:}\textcircled{r}\varphi}{\langle w, R \cup \{w\}, F - \{w\}\rangle^C{:}\varphi}
$$

where R is the set of remembered states and F the set of explicitly forgotten states. On the other hand, the rules for ($\textcircled{k}$) and ($\neg\textcircled{k}$) would be

$$(\textcircled{k})\ \frac{\langle w, \{v_1, \ldots v_k\}, F\rangle^C{:}\textcircled{k}}{w \approx v_1 \mid \cdots \mid w \approx v_k \mid \langle w, \emptyset, F\rangle^C{:}\textcircled{k}} \qquad (\neg\textcircled{k})\ \frac{\langle w, R, \{v_1, \ldots v_k\}\rangle^C{:}\neg\textcircled{k}}{w \approx v_1 \mid \cdots \mid w \approx v_k \mid \langle w, R, \emptyset\rangle^C{:}\neg\textcircled{k}}$$

Notice the symmetry between the rules, which corresponds to the symmetry in the semantic definition of ⓡ and ⓕ. Besides these changes, the tableau rules and the completeness argument remain roughly the same.

References

1. Areces, C.: Hybrid logics: The old and the new. In: Proc. of LogKCA-07, San Sebastian, Spain (2007)
2. Areces, C., Figueira, D., Figueira, S., Mera, S.: Expressive power and decidability for memory logics. In: Proc. of WoLLIC 2008. Volume 5110 of LNCS., Springer (2008) 56–68
3. Areces, C., Figueira, D., Figueira, S., Mera, S.: Expressive power and decidability for memory logics. Technical report, INRIA Nancy, Grand Est (2008) Extended version of [2].
4. Areces, C., Figueira, S., Mera, S.: Completeness results for memory logics. In: Proc. of LFCS 2009. Volume 5407 of LNCS., Springer (2009)
5. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press (2001)
6. Blackburn, P., Wolter, F., van Benthem, J., eds.: Handbook of Modal Logics. Elsevier (2006)
7. Areces, C., ten Cate, B.: Hybrid logics. [6] 821–868
8. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Kluwer academic publishers (2007)
9. Harel, E., Lichtenstein, O., Pnueli, A.: Explicit clock temporal logic. In: Proc. of LICS'90. (1990) 402–413
10. Plaza, J.: Logics of public communications. In: Proc. of 4th International Symp. on Methodologies for Intelligent Systems. (1989) 201–216
11. van Benthem, J.: Logics for information update. In: Proc. of TARK 2001, Morgan Kaufmann Pub. (2001) 51–67
12. van Benthem, J., van Eijck, J., Kooi, B.: Logics of communication and change. Information and Computation **204**(11) (2006) 1620–1662
13. Chandra, A., Merlin, P.: Optimal implementation of conjunctive queries in relational databases. In: Proc. of 9th ACM Symp. on Theory of Computing. (1977) 77–90
14. Bolander, T., Blackburn, P.: Termination for hybrid tableaus. Journal of Logic and Computation **17** (2007) 517–554
15. Papadimitriou, C.: Computational Complexity. Addison-Wesley (1994)
16. Franceschet, M., de Rijke, M.: Model checking hybrid logics (with an application to semistructured data). Journal of Applied Logic **4**(3) (2006) 279–304
17. Clarke, E., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Cambridge, Massachusetts (1999)