Towards a Reasoning Model for Context-aware Systems: Modal Logic and the Tree Model Property

Yensen Limón[1], Everardo Bárcenas[1,2], Edgard Benítez-Guerrero[1], Carmen Mezura-Godoy[1]

[1] Universidad Veracruzana, Facultad de Estadística e Informática, Mexico

[2] Consejo Nacional de Ciencia y Tecnología, Mexico

Abstract. Modal logics forms a family of formalisms widely used as reasoning frameworks in diverse areas of computer science. Description logics and their application to the web semantic is a notable example. Also, description logics have been recently used as a reasoning model for context-aware systems. Most reasoning algorithms for modal (description) logics are based on tableau constructions. In this work, we propose a reasoning (satisfiability) algorithm for the multi-modal K_m with converse. The algorithm is based on the finite tree model property and a Fischer-Ladner construction. We show the algorithm is sound and complete, and we provide the corresponding complexity analysis. We also present some exploratory results of a preliminary implementation of the algorithm.

Keywords: Modal logic, context-aware systems, reasoning.

1 Introduction

Modal logics forms a family of formalism including temporal, dynamic, epistemic and description logics. These formalisms have been widely used as reasoning frameworks in diverse areas of computer science, such as artificial intelligence, databases, program analysis, distributed computing, etc. [17, 15, 10]. The application of description logics as knowledge representation framework (language and reasoning for ontologies) in the semantic web is a notable example [3]. In recent years, due to the well-known excellent balance between the expressive power of description logic and the computational complexity of the associated algorithms, context aware computing inference systems have been studied in the modal (descriptive) setting [8, 6]. However, current context aware systems, which are supposed to efficiently interact with a legion of context variables, still demands more expressive power without performance detriment [8]. Motivated by the development of expressive context aware inference systems, as an starting point, we propose in the current work a reasoning algorithm for the multimodal logic

K_m with converse, which can be seen as a syntactic variant of the description logic $\mathcal{ALCI}$.

The basic modal logic K can be syntactically introduced as classical propositional logic extended with constructors for expressing modalities, such as possibility and necessity. From the seminal work of van Benthem [7], we know modal logic forms an important fragment of first order logic. The importance of this fragment comes from the associated well known and behaved computational properties, such as model checking and satisfiability. A possible explanation of this nice and robust computational behavior, provided by van Benthem [7], is that modal logic cannot distinguish bisimilar models. Another explanation comes from Vardi [20], and it concerns the tree model property of the logic. In the current work, we follow the second approach, more precisely, we propose a satisfiability algorithm for modal logic based on its finite tree model property. The algorithms actually builds candidate trees in the style of Fischer-Ladner [4, 5]. Moreover, the algorithm works for the modal logic extended with multimodalities K_m, including converse modalities, known in description and dynamic logics as inverse roles and programs, respectively.

We distinguish two types of techniques in the development of modal reasoners: purpose-built and translational. Most of purpose-built translational tools are based on tableau constructions [13, 12, 19, 11], however, there are other few methods, such as sequent calculus [21] and coalgebras [9]. In the translational methods, we may find translations to SAT solvers [18], to first order logic [14, 2], including SMT and QBF solvers [1, 16], respectively. In [16], it is also presented a translation of modal logic formulas to types, which can be seen as an on-the-fly construction of the corresponding automaton. For the current work, we focus on the purpose-built approach. In particular, the proposed algorithm is based in the finite tree model property of modal logic. More precisely, the algorithm performs a Fischer-Ladner construction of candidate trees. Although there are already highly optimized modal solvers successfully working in practice [13, 12], to the best of our knowledge, this is the first reasoning algorithm based on the finite tree model property.

The paper is organized as follows: in Section 2, we describe the multimodal logic K_m with converse; the finite tree model property of the logic is studied in Section 3; in Section 4, a satisfiability algorithm is described in terms of Fischer-Ladner constructions of trees, it is also shown the algorithm is correct, that is, sound and complete, and a complexity analysis is also provided; we conclude in Section 5 with a summary of the current work and a discussion on further research perspectives.

2 Preliminaries

In the current section, we define the propositional multimodal logic K_m with converse. Whereas syntax of the logic is defined as boolean combination of propositions and modal constructors, formula semantics is described in terms of Kripke (relational) structures.

2.1 Multimodal logic K_m with converse

We assume a fixed alphabet (P, M), where P is a countable set of proposition variables and M is a finite set of modalities. We also assume there is a bijection partitioning the set of modalities. We write $\overline{m}$ to denote the inverse of a modality m.

Definition 1 (Syntax)**.** The set of modal formulas is defined by the following grammar:

$$\phi := p \mid \neg\phi \mid \phi \vee \phi \mid \langle m \rangle \phi$$

where p ranges over propositions and m over modalities

Formulas are interpreted as subset nodes on Kripke structures, which can be intuitively seen as labeled directed graphs. Propositions serve as node labels, whereas negation and disjunction are interpreted as the complement and union of sets, respectively. Existential modal formulas $\langle m \rangle \phi$ holds in nodes able to access through m to a node belonging to the interpretation of ϕ.

Other classical syntactic sugar is also considered: tautologies $\top := \phi \vee \neg\phi$; contradictions $\bot := \neg\top$; conjunctions $\phi \wedge \psi := \neg(\neg\phi \vee \neg\psi)$; and universal modal formulas $[m]\,\phi := \neg\langle m \rangle \neg\phi$. These shorthands are interpreted as expected, in particular, universal modal formulas $[\phi]$ denote the nodes where *all* their accessible nodes through m support ϕ.

Before giving a formal semantics of modal formulae, we give a precise notion of Kripke structures.

Definition 2 (Kripke structure)**.** A Kripke structure is a tuple $K = (N, R, L)$, where: N is a non-empty and countable set of nodes; R is a family of binary relations $R^m : N \times N$, written $n \in R(n, m)$ for each modality m; and L is a left-total labeling function $L : N \mapsto 2^P$.

We now give a precise notion of formula semantics.

Definition 3 (Semantics)**.** Given a Kripke structure $K = (N, R, L)$, modal formulas are interpreted as follows:

$$\begin{aligned}
[\![p]\!]^K &= \{n \mid p \in L(n)\} \\
[\![\neg\phi]\!]^K &= N \setminus [\![\phi]\!]^K \\
[\![\phi \vee \psi]\!]^K &= [\![\phi]\!]^K \cup [\![\psi]\!]^K \\
[\![\langle m \rangle \phi]\!]^K &= \left\{n \mid R(n, m) \cap [\![\phi]\!]^K \neq \emptyset\right\}
\end{aligned}$$

We say a formula ϕ is satisfiable, if and only if, there is a Kripke structure K such that the interpretation of ϕ with respect to K is not empty, that is, $[\![\phi]\!]^K \neq \emptyset$. In such a case, we also say K is a model of ϕ. If any Kripke structure is a model of ϕ, we say ϕ is valid. We say two formulas ϕ and ψ are equivalent, if and only if, their interpretations coincide for any Kripke structure K, that is, $[\![\phi]\!]^K = [\![\psi]\!]^K$.

The negation normal form of a formula is an equivalent formula where negation only occurs in front of propositions. In order to give a precise notion of the negation normal form, we then consider the following definition.

Definition 4. Given a modal formula ϕ, its negation in normal form $\mathrm{nnf}(\phi)$ is inductively defined as follows:

$$\begin{aligned} \mathrm{nnf}(p) =& \neg p \\ \mathrm{nnf}(\phi \vee \psi) =& \mathrm{nnf}(\phi) \wedge \mathrm{nnf}(\psi) \\ \mathrm{nnf}\left(\langle m \rangle \phi\right) =& [m]\, \mathrm{nnf}(\phi) \end{aligned}$$

The negation normal form of a formula ϕ is defined by $\phi \left[{}^{\mathrm{nnf}(\psi)}/_{\neg\psi}\right]$. Then, in order to consider formulas in negation normal form only, we need to consider an extension of the logic with conjunctions and universal modal formulas with the expected semantics.

Proposition 1. *A formula and its negation normal form are equivalent.*

Hence, with out loss of generality, we consider now formulas in negation normal form.

3 The Tree Model Property

In this Section, we provide a description of the finite tree model property: if a formula is satisfiable, then it is also satisfiable in a finite tree shaped Kripke structure.

Definition 5. Given a formula ϕ and a Kripke structure $K = (N, R, L)$ satisfying ϕ in a node $n \in N$, we inductively define the following finite tree shaped Kripke structure $\sigma\left(\phi, K, n\right) = \left(\sigma\left(N\right), \sigma\left(R\right), \sigma L\right)$ as follows:

– the root is $\sigma(n) \in \sigma\left(N\right)$, in case $\sigma(n)$ has already been added in a previous step, then a fresh copy of $\sigma(n)$ is considered;
– in case ϕ is a proposition p, then $p \in \sigma\left(L(\sigma(n))\right)$;
– if ϕ is a negation $\neg p$, $p \notin \sigma\left(L(\sigma(n))\right)$;
– when ϕ is a disjunction $\psi \vee \varphi$, then we know that n satisfies ψ or φ, in case n satisfies ψ, then $\sigma\left(\phi, K, n\right)$ is defined by $\sigma\left(\psi, K, n\right)$, otherwise, it is defined by $\sigma\left(\varphi, K, n\right)$;
– if it is the case that ϕ is a conjunction $\psi \wedge \varphi$, then $\sigma\left(\phi, K, n\right)$ is defined by the two branched tree formed by $\sigma\left(\psi, K, n\right)$ and $\sigma\left(\varphi, K, n\right)$ with $\sigma(n)$ as the common root.
– if ϕ is an existential modal formula $\langle m \rangle \phi$, then $\sigma\left(\phi, K, n\right)$ is formed by $\sigma\left(\psi, K, n\right)$, such that for a node $n' \in R(n, m) \cap [\![\psi]\!]^K$, $\sigma\left(n'\right) \in \sigma(R(\sigma, m))$ and $\sigma\left(n'\right)$ is the root of $\sigma\left(\psi, K, n\right)$;
– the case of universal modal formulas $[m]\, \psi$ is similar as the previous one, but the construction is done with respect to each node $n' \in R(n, m) \cap [\![\psi]\!]^K$.

Theorem 1 (Finite tree model property). *For any formula ϕ and Kripke structure K, we have that*

$$n \in [\![\phi]\!]^K \text{ if and only if } \sigma(n) \in [\![\phi]\!]^{\sigma(\phi,K,n)}$$

By Definition 5, the proof goes smoothly by structural induction on ϕ. Also, it is clear that $\sigma(\phi, K, n)$ is finite and tree shaped.

4 Satisfiability

In this Section, we describe a satisfiability algorithm for the multi-modal K_m with converse, that is, given a formula, the algorithm decides whether or not the input formula is satisfiable. Recall that if a formula is satisfiable, then the formula is satisfiable by a tree shaped Kripke structure. Then, the algorithm builds candidate trees in a bottom-up manner: starting from the leaves, the algorithm adds parents iteratively and consistently until a satisfying tree is found. The representation of trees is in the style of Fischer-Ladner [4, 5]. Before defining the algorithm, we first need some notation.

Definition 6. . We define the following binary relation R^{FL} on formulas with $i = 1, 2$ as follows:

$$R^{FL}(\phi_1 \circ \phi_2, \phi) \qquad R^{FL}(\langle m\rangle \phi, \phi) \qquad R^{FL}([m]\phi, \phi)$$

where $\circ \in \{\wedge, \vee\}$ and $i = 1, 2$.

We now define the Fischer-Ladner closure.

Definition 7 (Fischer Ladner Closure)**.** Given a formula ϕ, the Fischer-Ladner closure of ϕ, written $FL(\varphi)$, is defined as $F_k(\phi)$ for the smallest k, such that $F_k(\phi) = F_{k+1}(\phi)$, where

$$FL_0(\varphi) = \{\varphi\}$$
$$FL_{i+1}(\varphi) = FL_i(\varphi) \cup \{\psi' \mid R^{FL}(\psi, \psi'), \psi \in FL_i(\varphi)\}$$

for $i > 0$.

The Fischer-Ladner representation of leaves, and hence trees, is based in the lean set, which is now defined.

Definition 8 (Lean)**.** Given a formula ϕ, its lean is defined as the set composed by proposition and modal subformulas of ϕ, together with formulas $\langle m\rangle \top$, for every m occurring in ϕ, and p' which is a proposition not occurring in ϕ. More precisely,

$$lean(\varphi) = \{p, \langle m\rangle \psi, [m]\psi \in FL(\varphi)\} \cup \{\langle m\rangle\top, p'\}$$

We are now ready to define the set of nodes in Fischer-Ladner trees.

Algorithm 1 Satisfiability algorithm.

```
$Y \leftarrow N^{\phi}$
$X \leftarrow Leaves(Y)$
$X_0 \leftarrow \emptyset$
while $X \neq X_0$ do
    if $X \vdash \phi$ then
        return true
    end if
    $X_0 \leftarrow X$
    $(X, Y) \leftarrow Update(X, Y)$
end while
return false
```

Definition 9 (Nodes)**.** Given a formula ϕ, a ϕ-node, or simply a node, is defined as a lean subset with the following constraints:

– at least one proposition occurs in it; and
– if $\langle m \rangle \psi$ occurs, then $\langle m \rangle \top$ also does.

The set of nodes corresponding to a given formula ϕ is written N^{ϕ}.

Definition 10 (Fischer-Ladner tree)**.** Given a formula ϕ, a ϕ-tree T, or simply a tree, is inductively defined as follows:

– the empty set is a tree;
– the tuple $(n, T_1, \ldots, T_n)$ is a tree, provided that n is a node, called the root, and T_i $(i = 1, \ldots, n)$ are trees.

The algorithm corresponding to the satisfiability of formulas is defined in Algorithm 1.

We now give a precise description of each notion involved in the algorithm.

We first define the set of leaves as the trees $(n, \emptyset)$, where n does not contain formulas of the form $\langle m \rangle \phi$. Then, $Leaves(N)$ for a set of nodes N, contains all the leaves in N.

We now define the entailment relation between nodes and formulas.

Definition 11. Given a formula ϕ and a node n, we inductively define the entailment relation $n \vdash$ as follows:

$$\frac{\phi \in n}{n \vdash \phi} \qquad \frac{p \notin n}{n \vdash \neg p} \qquad \frac{}{n \vdash \top}$$

$$\frac{n \vdash \phi \text{ and } n \vdash \psi}{n \vdash \phi \wedge \psi} \qquad \frac{n \vdash \phi \text{ or } n \vdash \psi}{n \vdash \phi \vee \psi}.$$

Abusing of notation, we extend the notion of entailment between trees $T = (n, T_1, \ldots, T_n)$ and formulas ϕ, written $T \vdash \phi$, when $n \vdash \phi$. This notion is also extended to set of trees X, written $X \vdash \phi$, when there is a tree $T \in X$, such that $T \vdash \phi$. The complement of this relation $\not\vdash$ is defined as expected.

The $\mathit{Update}(X, Y)$ function, for a set of trees X and a set of nodes Y, adds parents in Y to trees in X in a consistent manner. This results in a pair (X', Y'), where X' is the new set of trees and Y' is the new set of nodes (the nodes used as parents are removed).

Definition 12. Given a formula ϕ, a set of trees X and a set of nodes Y, we define the *Update* function as follows

$$\mathit{Update}(X, Y) = (X', Y')$$

where

$$X' = \{(n, T_1, \ldots, T_n) \mid n \in Y, \varDelta_m(n, T_i)\}$$

for each $i = 1, \ldots, n$ and m occurring in ϕ, and where

$$\varDelta_m(n, T) = \begin{cases} \textbf{true} & \text{if } \forall \langle m \rangle \psi \in \mathit{lean}(\phi) : \langle m \rangle \psi \in n \text{ iff } n' \vdash \psi \\ & \forall [m] \psi \in \mathit{lean} : [m] \phi, \langle m \rangle \top \in n \text{ iff } n' \vdash \psi \\ & [m] \psi \in n, \langle m \rangle \top \notin n \text{ iff } n' \nvdash \psi \\ \textbf{false} & \text{otw.} \end{cases}$$

where n' is the root of T, and

$$Y' = Y \setminus \{n \mid \text{ n is the root of tree in } X'\}$$

Notice that *Update* is a monotone function, hence, it has a fixed-point. We now show the algorithm is correct. This is shown through soundness and completeness.

Theorem 2 (Soundness). *Given a formula ϕ, if the algorithm returns true, then ϕ is satisfiable.*

Proof. If the algorithm returns true, we know there is a tree $T = (n, T_1, \ldots, T_n)$ entailing ϕ, that is, $T \vdash \phi$. We now construct a tree shaped Kripke structure $K = (N, R, L)$ from T, satisfying ϕ.

- N is composed by each node in T;
- for each subtree $T' = (n', T'_1, \ldots, T'_k)$ in T and for each $i = 1, \ldots, k$, $n'' \in R(n', m)$, such that n'' is the root of T'_i and $\varDelta_m(n', n'')$; and
- for each node n' in N and each proposition $p \in \mathit{lean}(\phi)$, if $p \in n'$, then $p \in L(n')$.

That K satisfies ϕ is proven by a straightforward structural induction, due to the soundness of relations $\varDelta$ and $\vdash$. □

Theorem 3 (Completeness). *If a satisfiable formula ϕ is given to the algorithm, then the algorithm returns true.*

Proof. By Theorem 1, we know there is a tree shaped Kripke structure $K = (N, R, L)$ satisfying ϕ. Moreover, K is defined as described in Definition 5.

We first show a tree T isomorphic to K entails ϕ. For this, we first define T as follows:

– for each each node n in N, there is a corresponding node $\tau(n)$ in T;
– for each $n_1, n_2, \ldots, n_k \in R(n, m)$ for any m occurring in ϕ, $(\tau(n), T_1, T_2, \ldots, T_k)$ is a subtree in T, such that $\tau(n_i)$ is the root of T_i for $i = 1, \ldots, k$;
– for each $p \in L(n)$, $p \in \tau(n)$;
– for each $\langle m \rangle \psi, \in \mathit{lean}(\phi)$, if $n \in [\![\langle m \rangle \psi]\!]^K$, then $\langle m \rangle \psi \in \tau(n)$, the same applies for formulas $[m]\,\psi$ in the lean; and finally,
– for each $\tau(n)$ in T, if $\langle m \rangle \psi \in \tau(n)$, then $\langle m \rangle \top \in \tau(n)$.

By a straightforward structural induction on ϕ, it is shown that $T \vdash \phi$ (recall ϕ is satisfied in the root of K).

We now show T is constructed by the algorithm. This is shown by induction on the height of K. It is clear T has the same height of K. The base case is immediate. We now assume K has has height n, then the algorithm has constructed trees $T_1, T_2, \ldots, T_k$ corresponding to the subtrees of K. Now notice that if $n > 1$, then ϕ is of the form $\langle m \rangle \psi$ or $[m]\,\psi$. Hence, considering n is the root of K and that it is the one satisfying ϕ, then node $\tau(n)$ is still in Y. By completeness of relation Delta, we know then for each m occurring in ϕ and each $i = 1, \ldots, k$, $\varDelta_m(n, n_i)$, provided that n_i is the root of T_i. We then conclude $\tau(n) \vdash \phi$. □

Theorem 4 (Complexity). *The satisfiability algorithm is in EXPTIME.*

Proof. First notice that the lean set has linear size with respect to the input formula ϕ. Hence, there is an exponential number of nodes. Each node is at most of the same size than ϕ. Hence, for each node n, $n \vdash \phi$ takes linear time. When it comes to trees, $\vdash$ is clearly at most exponential. Testing the set of trees also takes at most exponential: it takes the sum of exponentials searches. Finally, the exponential bound on the *Update* function comes from the exponentially bounded size of its search space (nodes and trees), and from the fact that $\varDelta$ has linear cost. □

We are currently implementing a preliminary version of the algorithm in Java language. Some exploratory results of this preliminary version are depicted in Figure 1. We tested the implementation in a computer with the following features: Windows 8 operating system, AMD processor A6 2.7GHz., 8Gb of RAM. In this preliminary version, we have implemented the set of nodes in an explicit way, which results very expensive in practice. In Figure 1, it is easy to notice that incrementing a single modal level in the input formula, which implies a higher tree, drastically impact in the performance of the algorithm. In order to alleviate this issue, we plan to soon incorporate to the algorithm a non-explicit representation of nodes, such as Binary Decision Diagrams [16].

5 Conclusions

In this paper, we introduced a satisfiability algorithm of the multi-modal logic K_m with converse. The algorithm is based on the finite tree model property

Formula	Time (milliseconds)
$\langle 1 \rangle a$	2153
$\langle 1 \rangle a \wedge b$	5180
$\langle 1 \rangle a \wedge \langle 2 \rangle b$	11419
$\langle 1 \rangle a \wedge \langle \overline{1} \rangle b$	9188
$\langle 1 \rangle a \wedge \langle \overline{2} \rangle b$	9110
$\langle 1 \rangle a \wedge \langle 2 \rangle b \wedge c$	42604
$\langle 1 \rangle a \wedge \langle 2 \rangle b \wedge \neg c$	41403
$\langle 1 \rangle a \wedge \langle 2 \rangle b \wedge \neg(c \vee d)$	38937
$\langle 1 \rangle \langle 1 \rangle a$	157359
$\langle 1 \rangle \langle 1 \rangle a \wedge b$	318714
$\langle 1 \rangle \langle 1 \rangle a \wedge \langle 1 \rangle b$	863249
$\langle 1 \rangle \langle 1 \rangle a \wedge \langle \overline{1} \rangle b$	595539
$\langle 1 \rangle \langle 1 \rangle a \wedge \langle 2 \rangle b$	897349
$\langle 1 \rangle \langle 1 \rangle a \wedge \langle \overline{2} \rangle b$	904157
$\langle 1 \rangle \langle 1 \rangle a \wedge \langle 2 \rangle \langle 2 \rangle b$	2043845

Fig. 1. Results of a preliminary implementation of the satisfiability algorithm.

of the logic. We also showed the algorithm is sound and complete, and that it takes exponential time. Some exploratory results of a naive and non-optimized implementation of the algorithm were also described.

We are currently implementing non-explicit representations of the set of nodes. In particular, we are implementing a BDD-based version the algorithm [16]. We plan to extend the current algorithm, as described in [4, 5], to more expressive logics, such as the μ-calculus with arithmetic constraints. We are also studying the Description Logics counterpart of these expressive logics. This is with the final aim to provide an efficient and expressive reasoning framework for context-aware systems [6, 8].

Acknowledgment. This work was partially developed under the support of the Mexican National Science Council (CONACYT) in the scope of the Cátedras CONACYT project Infraestructura para Agilizar el Desarrollo de Sistemas Centrados en el Usuario (Ref 3053).

References

1. Areces, C., Fontaine, P., Merz, S.: Modal satisfiability via SMT solving. In: Nicola, R.D., Hennicker, R. (eds.) Software, Services, and Systems - Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering. Lecture Notes in Computer Science, vol. 8950, pp. 30–45. Springer (2015)
2. Areces, C., Gennari, R., Heguiabehere, J., de Rijke, M.: Tree-based heuristics in modal theorem proving. In: ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence. pp. 199–203 (2000)
3. Baader, F., Hollunder, B.: KRIS: knowledge representation and inference system. SIGART Bulletin 2(3), 8–14 (1991)

4. Bárcenas, E., Genevès, P., Layaïda, N., Schmitt, A.: Query reasoning on trees with types, interleaving, and counting. In: Walsh, T. (ed.) IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 718–723. IJCAI/AAAI (2011)
5. Bárcenas, E., Lavalle, J.: Global numerical constraints on trees. Logical Methods in Computer Science 10(2) (2014)
6. Benítez-Guerrero, E.: Context-aware mobile information systems: Data management issues and opportunities. In: Arabnia, H.R., Hashemi, R.R., Vert, G., Chennamaneni, A., Solo, A.M.G. (eds.) Proceedings of the 2010 International Conference on Information & Knowledge Engineering. pp. 127–133. CSREA Press (2010)
7. van Benthem, J.: Modal Logic and Classical Logic. Bibliopolis (1983)
8. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2), 161–180 (2010)
9. Calin, G., Myers, R.S.R., Pattinson, D., Schröder, L.: Coloss: The coalgebraic logic satisfiability solver. Electr. Notes Theor. Comput. Sci. 231, 41–54 (2009)
10. Fitting, M.: Modality and databases. In: Dyckhoff, R. (ed.) Automated Reasoning with Analytic Tableaux and Related Methods, International Conference. Lecture Notes in Computer Science, vol. 1847, pp. 19–39. Springer (2000)
11. Genevès, P., Layaïda, N., Schmitt, A., Gesbert, N.: Efficiently deciding μ-calculus with converse over finite trees. ACM Trans. Comput. Log. 16(2), 16 (2015)
12. Haarslev, V., Möller, R.: RACER system description. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) Automated Reasoning, First International Joint Conference. Lecture Notes in Computer Science, vol. 2083, pp. 701–706. Springer (2001)
13. Horrocks, I., Patel-Schneider, P.F.: Optimizing description logic subsumption. J. Log. Comput. 9(3), 267–293 (1999)
14. Hustadt, U., Schmidt, R.A., Weidenbach, C.: MSPASS: subsumption testing with SPASS. In: Lambrix, P., Borgida, A., Lenzerini, M., Möller, R., Patel-Schneider, P.F. (eds.) Proceedings of the 1999 International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 22. CEUR-WS.org (1999)
15. Kraus, S., Lehmann, D.J.: Knowledge, belief and time. Theor. Comput. Sci. 58, 155–174 (1988)
16. Pan, G., Sattler, U., Vardi, M.Y.: Bdd-based decision procedures for the modal logic K. Journal of Applied Non-Classical Logics 16(1-2), 169–208 (2006)
17. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science. pp. 46–57. IEEE Computer Society (1977)
18. Sebastiani, R., Vescovi, M.: Automated reasoning in modal and description logics via SAT encoding: the case study of k(m)/alc-satisfiability. J. Artif. Intell. Res. (JAIR) 35, 343–389 (2009)
19. Tanabe, Y., Takahashi, K., Hagiya, M.: A decision procedure for alternation-free modal μ-calculi. In: Areces, C., Goldblatt, R. (eds.) Advances in Modal Logic 7. pp. 341–362. College Publications (2008)
20. Vardi, M.Y.: Why is modal logic so robustly decidable? In: Immerman, N., Kolaitis, P.G. (eds.) Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 31, pp. 149–184. American Mathematical Society (1996)
21. Voronkov, A.: How to optimize proof-search in modal logics: new methods of proving redundancy criteria for sequent calculi. ACM Trans. Comput. Log. 2(2), 182–215 (2001)