# Model Checking with Multi-Valued Logics

Glenn Bruns Patrice Godefroid

Bell Laboratories, Lucent Technologies, {grb,god}@bell-labs.com

**Abstract.** In multi-valued model checking, a temporal logic formula is interpreted relative to a structure not as a truth value but as a lattice element. In this paper we present new algorithms for multi-valued model checking. We first show how to reduce multi-valued model checking with any distributive DeMorgan lattice to standard, two-valued model checking. We then present a direct, automata-theoretic algorithm for multi-valued model checking with logics as expressive as the modal mu-calculus. As part of showing correctness of the algorithm, we present a new fundamental result about extended alternating automata, a generalization of standard alternating automata.

## 1 Introduction

In multi-valued model checking, one interprets a temporal logic formula on a multi-valued Kripke structure, which is like a Kripke structure except that an atomic proposition is interpreted at a state as a lattice element, not a truth value. The meaning of a temporal logic formula at a state in such a structure is then also given as a lattice element.

Multi-valued model checking is proving valuable as the basis for a variety of new verification methods. For example, the abstraction method of [4] involves model checking with the lattice $L_3$ of Figure 1, where 1 represent truth, 0 represents falsity, and 1/2 represents "unknown whether true or false". Model checking with the lattice $L_{2,2}$ can be used to analyze whether conflict will arise when multiple requirements are combined [8, 18]. Temporal logic query checking [6, 3, 9] can be regarded as model checking over lattices in which each element is a set of propositional formulas.

One approach to multi-valued model checking is the *reduction method*, in which a multi-valued model checking problem is reduced to a set of standard, two-valued model checking problems [2, 19, 18]. For example, in the case of lattice $L_3$, a model checking problem for a Kripke structure over $L_3$ can be reduced to two model checking problems for Kripke structures over $L_2$. Another approach is the *direct method*, in which multi-valued model checking is performed directly using special-purpose algorithms. An advantage of the reduction method is that it can use existing tools, and benefits as these tools are improved. The advantage of the direct approach is that it works in a more "on-demand" manner than the reduction approach (more comparisons are made in Section 6).

This paper describes improved reduction and direct methods for multi-valued model checking. A problem with existing reduction methods [2, 19] is their limitation to selected sub-classes of DeMorgan lattices. A recent method [17] is more

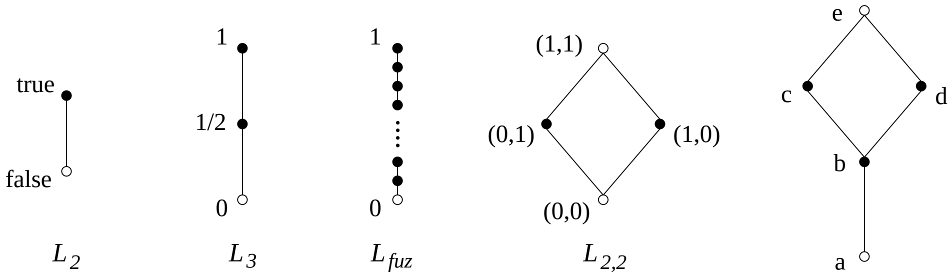


**Fig. 1.** Some distributive lattices

general but also more complicated, involving a step that uses an additional logic. Our method is simple and general. We show that, for a finite distributive lattice, the number of standard model checks required is equal to the number of join-irreducible elements of the lattice in the worst case. From a multi-valued Kripke structure over a finite distributive lattice, we show how a standard Kripke structure can be derived for each join-irreducible element of the lattice, and how the results of model checking on each of these Kripke structures can be combined to give a result for the multi-valued model check. The method yields complexity bounds for the multi-valued model-checking problem for various temporal logics.

Existing work on direct methods is limited in the class of lattices that are handled, or the logic that is supported. In [4] an algorithm is defined for CTL over $L_3$. In [10] an automata-theoretic algorithm is defined for LTL over finite linear orders. In [7] a BDD-based algorithm is defined for CTL over DeMorgan lattices. Our method is automata-theoretic and handles all DeMorgan lattices and the full modal mu-calculus. To adapt the automata-theoretic method to multi-valued model checking, we use extended alternating automata (EAA) [3], which extend alternating automata (AA). In model checking applications of AA (e.g., [21]), an input tree of the automaton has nodes that are labelled with sets of atomic propositions, and a run of the automaton has no value associated with it. With EAA, the nodes of the input tree are labelled with functions mapping atomic propositions to elements of a lattice, and a run has an associated value. We show how to use EAA for multi-valued model checking, but also prove a fundamental result about EAA that is interesting independently of this application: that the set of values of all the accepting runs of an EAA has a maximal element.

The following section briefly covers some background material. In Section 3, we define our reduction method. In Section 4 we define extended alternating automata, and in Section 5 we show how to directly model check with them. We conclude in Section 6 by comparing the reduction and direct approaches to multi-valued model checking.

## 2 Background

**Lattices and Negation**. We take for granted the notion of lattice and complete lattice. We write $x \vee y$ or $\bigvee P$ for join and $x \wedge y$ or $\bigwedge P$ for meet (where $P$ is a set). Every complete lattice has a greatest element, called *top*, and a least element, called *bottom* (and written $\perp$). Every finite lattice is complete. A lattice is *distributive* if $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ for all lattice elements $x, y, z$.

A *join-irreducible element* $x$ of a distributive lattice $L$ is an element that is not bottom and for which $x = y \vee z$ implies $x = y$ or $x = z$. If $L$ is finite, the join-irreducible elements are easily spotted in the Hasse diagram for $L$ as elements having exactly one lower cover (i.e. one line connected to the element from below). The darkened elements in Figure 1 are the join-irreducible ones. We write $\mathcal{J}(L)$ for the set of all join-irreducible elements of $L$.

If one orders truth and falsity as shown in lattice $L_2$ of Figure 1, then conjunction can be interpreted as meet and disjunction as join. In this way conjunction and disjunction can be interpreted over an arbitrary lattice. To interpret negation on lattices, a restricted class of lattices must be used if one hopes to obtain expected properties of negation. *Boolean* lattices support a strong sense of complement. Every element $x$ in such a lattice has a unique complement $\neg x$ such that $x \vee \neg x$ equals the top element of the lattice and $x \wedge \neg x$ equals the bottom element of the lattice. Lattice $L_2$ of Fig. 1 is boolean. However, there are "few" boolean lattices.

In a *DeMorgan* (or *quasi-boolean*) lattice [1], every element $x$ has a unique complement $\neg x$ such that $\neg\neg x = x$, DeMorgan's laws hold, and $x \leq y$ implies $\neg y \leq \neg x$. DeMorgan lattices can be characterized as lattices with horizontal symmetry [7]. Lattice $L_3$ of Fig. 1 is DeMorgan, but not boolean. Using DeMorgan complement we get that $\neg 0 = 1$, $\neg 1/2 = 1/2$, and $\neg 1 = 0$

A *Heyting algebra* is a lattice with a bottom element in which every element $x$ has a unique *relative pseudo-complement* $\neg x$ defined as the greatest element $y$ such that $x \wedge y$ equals the lattice's bottom element. In the case of finite lattices, Heyting algebras and distributive lattices are the same thing [13]. The rightmost lattice in Fig. 1 is a Heyting algebra but is not DeMorgan. In this lattice, using relative pseudo-complement as complement, we get $\neg a = e$ and $\neg b = a$. In lattice $L_3$ we get $\neg 0 = 1$, $\neg 1/2 = 0$, and $\neg 1 = 0$. Some DeMorgan lattices are not Heyting algebras.

Reasoning about partial information with three-valued logic based on $L_3$ is an important application of multi-valued model checking, and since in this application we want to interpret negation in the DeMorgan sense, we adopt DeMorgan lattices for multi-valued model checking.

**The Modal Mu-Calculus**. The modal mu-calculus [20] is an expressive modal logic that includes as fragments linear-time temporal logic (LTL) and computation-tree logic (CTL) [12]. Without loss of generality, we use a positive form of the modal mu-calculus in which negation applies only to atomic propositions. Formulas have the following abstract syntax, where $p$ ranges over a set $P$ of atomic propositions and $X$ ranges over a set $Var$ of fixed-point variables:

$$\phi ::= p \mid \neg p \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Box\, \phi \mid \Diamond\, \phi \mid X \mid \nu X.\phi \mid \mu X.\phi$$

In fixed-point formulas $\nu X.\phi$ and $\mu X.\phi$ the operators $\nu$ and $\mu$ bind free occurrences of $X$ in $\phi$. We call this logic $\mu L$.

A *Kripke structure* $M = (S, s_0, \Theta, \mathcal{R})$ consists of a set $S$ of states, an initial state $s_0$ in $S$, a mapping $\Theta$ from states to subsets of $P$, and a transition relation

$\mathcal{R} \subseteq S \times S$, assumed to be total. We say $M$ is *finite* if it has finitely many states. We write $s \to s'$ if $(s, s') \in \mathcal{R}$ and write $succ_{\mathcal{R}}(s)$ for the set $\{s' \in S \mid s \to s'\}$. For a finite subset $D$ of $I\!N$, we say $M$ *has degrees in* $D$ if $|succ_{\mathcal{R}}(s)| \in D$ for all states $s$ of $S$.

A Kripke structure $M = (S, s_0, \Theta, \mathcal{R})$ over a lattice $L$ differs from a standard Kripke structure in that now $\Theta$ maps a state to a mapping from propositions to elements of $L$. We write $P \to L$ for the set of all mappings from $P$ to $L$.

A *valuation* $\mathcal{V}$ over a lattice $L$ maps a variable to a mapping from states to elements of $L$. We write $()$ for the valuation such that $()(X)(s) = \bot$ for all $X$ and $s$ (it is required here that $L$ has a bottom element), and write $\mathcal{V}[X := f]$ for the valuation that is like $\mathcal{V}$ except that it maps $X$ to $f$.

We define the meaning $\|M, \phi\|_{\mathcal{V}}$ of a $\mu L$ formula relative to a Kripke structure $M = (S, s_0, \Theta, \mathcal{R})$ over lattice $L$ as a mapping from $S$ to $L$. In the following definition the function $f : (S \to L) \to (S \to L)$ is defined by $f(g) = \|M, \phi\|_{\mathcal{V}[X:=g]}$, and $\nu f$ and $\mu f$ stand for the greatest and least fixed-points of $f$. We know $f$ has greatest and least fixed-points by the Knaster-Tarski fixpoint theorem [23] because the functions in $S \to L$, under pointwise ordering, form a complete lattice, and function $f$ preserves this ordering.

**Definition 1.** *The interpretation $\|M, \phi\|_{\mathcal{V}}$ of a $\mu L$ formula relative to Kripke structure $M = (S, s_0, \Theta, \mathcal{R})$ and valuation $\mathcal{V}$ over complete DeMorgan lattice $L$ is defined as follows:*

$$
\begin{array}{rlrl}
\|M, p\|_{\mathcal{V}} &= \lambda s.\Theta(s)(p) & \|M, \phi_1 \wedge \phi_2\|_{\mathcal{V}} &= \lambda s.\|M, \phi_1\|_{\mathcal{V}}(s) \wedge \|M, \phi_2\|_{\mathcal{V}}(s) \\
\|M, \neg p\|_{\mathcal{V}} &= \lambda s.\neg\Theta(s)(p) & \|M, \phi_1 \vee \phi_2\|_{\mathcal{V}} &= \lambda s.\|M, \phi_1\|_{\mathcal{V}}(s) \vee \|M, \phi_2\|_{\mathcal{V}}(s) \\
\|M, \nu X.\phi\|_{\mathcal{V}} &= \nu f & \|M, \Box\, \phi\|_{\mathcal{V}} &= \lambda s.\bigwedge\{\|M, \phi\|_{\mathcal{V}}(s') \mid s \to s'\} \\
\|M, \mu X.\phi\|_{\mathcal{V}} &= \mu f & \|M, \Diamond\, \phi\|_{\mathcal{V}} &= \lambda s.\bigvee\{\|M, \phi\|_{\mathcal{V}}(s') \mid s \to s'\} \\
\|M, X\|_{\mathcal{V}} &= \mathcal{V}(X) & &
\end{array}
$$

If $\phi$ is a closed formula then we write $[(M, s), \phi]$ for the value $\|M, \phi\|_{()}(s)$ of formula $\phi$ at state $s$ of Kripke structure $M$. Given $\phi$, $(M, s)$, and $L$, computing $[(M, s), \phi]$ is called the *multi-valued model-checking problem.* If $M$ is a Kripke structure over lattice $L_2$, then we write $(M, s) \models \phi$ if $[(M, s), \phi] = \mathit{true}$.

**Proposition 1.** *The $\mu L$ semantics of Def. 1 collapses to the standard two-valued semantics of $\mu L$ when lattice $L$ is $L_2$ of Fig. 1.*

## 3 Reduction to 2-Valued Model Checking

In this section we show how multi-valued model checking of a $\mu L$ formula $\phi$ relative to a Kripke structure $M$ over a finite distributive lattice $L$ can be performed by model checking $\phi$ relative to a set of standard Kripke structures.

A key part of our approach is the treatment of negation. We transform $\phi$ to a formula $\phi'$ containing no negation symbols. Each negated proposition $\neg p$ in $\phi$ is replaced by $\tilde{p}$, where $\tilde{p}$ is a fresh proposition not already appearing in $\phi$. Correspondingly, $M$ is transformed to $M'$ by extending the proposition valuation $\Theta$ of $M$ to $\Theta'$, where $\Theta'(s)(\tilde{p}) = \neg\Theta(s)(p)$. Then $[(M, s), \phi] = [(M', s), \phi']$ for all states $s$ of $M$. In the rest of this section we consider only formulas of $\mu L$ not containing the negation symbol. Note that our step of eliminating negation symbols requires a negation operation on the underlying lattice.

### 3.1 Reduction Method

We now describe how to derive a standard Kripke structure $M_x$ from a Kripke structure $M$ over lattice $L$. If $M$ is defined to be $(S, s_0, \Theta, \mathcal{R})$, and $x$ is an element of $L$, then $M_x$ is defined to be $(S, s_0, \Theta_x, \mathcal{R})$, where

$$\Theta_x(s)(p) = \Theta(s)(p) \geq x$$

$M_x$ differs from $M$ only in its treatment of atomic propositions. In $M_x$, propositions with value $x$ or greater are regarded as true, and all others as false. Thus, if $x \geq x'$, we expect a formula that holds in $M_x$ to also hold in $M_{x'}$.

**Proposition 2.** *Let $M$ be a Kripke structure over a finite distributive lattice $L$, with $s$ in $M$ and $x, x'$ in $L$. Then $((M_x, s) \models \phi$ and $x \geq x') \Rightarrow (M_{x'}, s) \models \phi$*

The value of a formula relative a Kripke structure over a lattice $L$ can be determined by checking the standard Kripke structures derived from the join-irreducible elements of $L$.

**Lemma 1.** *Let $M$ be a Kripke structure over a finite distributive lattice $L$, with $s$ in $M$ and $x$ in $\mathcal{J}(L)$. Then $(M_x, s) \models \phi \Leftrightarrow x \leq [(M, s), \phi]$.*

From this lemma our main theorem follows using Birkhoff's representation theorem for finite distributive lattices, which states that every element $a$ of such a lattice can be represented as the join of all the join-irreducible elements less than or equal to $a$ in the lattice.

**Theorem 1.** *Let $M$ be a Kripke structure over a finite distributive lattice $L$, with $s$ in $M$. Then $[(M, s), \phi] = \bigvee\{x \in \mathcal{J}(L) \mid (M_x, s) \models \phi\}$.*

For example, consider the model checking of a formula $\phi$ relative to a structure $M$ over lattice $L_3$ of Fig. 1. The join-irreducible elements of $L_3$ are 1/2 and 1. Intuitively, the model $M_1$ represents a pessimistic view in which 1/2 is taken as false, while $M_{1/2}$ represents an optimistic view in which 1/2 is taken as true. The algorithm first checks whether $\phi$ holds in $M_1$. If so, the result is $\bigvee\{1/2, 1\}$, or 1. If not, it checks whether $\phi$ holds of model $M_{1/2}$. If so, the result is $\bigvee\{1/2\}$, or 1/2. Otherwise the result is $\bigvee \emptyset$, or 0.

Since two-valued model checking is a special case of multi-valued model checking, our reduction immediately gives the following complexity bounds for the multi-valued model-checking problem.

**Theorem 2.** *Let $L$ be a finite distributive DeMorgan lattice with $n$ join-irreducible elements, and let $TL$ denote $\mu L$ or any of its fragments. Then the multi-valued model-checking problem for $TL$ with respect to $L$ can be solved in time linear in $n$. Moreover, the complexity of multi-valued model checking for $TL$ has the same time and space complexity, both in the size of the Kripke structure and of the formula, as traditional two-valued model checking for $TL$.*

The linear complexity in the number of join-irreducible elements can be improved for some classes of lattices. For example, when the join-irreducible elements of a lattice $L$ are linearly ordered, a binary search (i.e., checking first the join-irreducible element in the middle of the lattice, then the join-irreducible element in the middle of the upper or lower half, etc.) can be performed instead of a linear search, providing a decision procedure for the multi-valued model-checking problem for $L$ with a worst-case time complexity of $O(log(n))$ instead of $O(n)$.

### 3.2 Multi-Valued Transitions

In Kripke structures with multi-valued transitions, transitions are represented by a function $R$ that maps pairs of states to lattice values. The $\mu L$ semantics (see Section 2) changes only for the modal operators, as follows:

$$\|M, \Box\, \phi\|_{\mathcal{V}} = \lambda s. \bigwedge\{\neg R(s, s') \vee \|M, \phi\|_{\mathcal{V}}(s') \mid \text{all } s'\}$$
$$\|M, \Diamond\, \phi\|_{\mathcal{V}} = \lambda s. \bigvee\{R(s, s') \wedge \|M, \phi\|_{\mathcal{V}}(s') \mid \text{all } s'\}$$

A Kripke structure with multi-valued transitions can be transformed to a structure without multi-valued transitions using the idea described in Definitions 16 and 17 of [16]. However, this transformation may in the worst case involve a blow-up of size $|L|$. Therefore we extend our reduction method to handle multi-valued transitions directly, with no blow-up in $|L|$. The extended method works in two steps. First, as before, from the original Kripke structure $M$ over a lattice $L$, we obtain a set $\{M_x \mid x \in \mathcal{J}(L)\}$ of structures. However, each structure $M_x$ now has two transition relations: $\mathcal{R}^+$ and $\mathcal{R}^-$. In the second step, each $M_x$ is translated to a standard Kripke structure $M'_x$ having only a single transition relation.

We now briefly cover the details. Suppose $M = (S, s_0, \Theta, R)$ is a Kripke structure over a finite distributive lattice $L$, where $R : S \times S \to L$ is the multi-valued transition function. Given a join-irreducible element $x$ of $L$, we define $M_x$ as before, except that now $M_x$ has the form $(S, s_0, \Theta_x, \mathcal{R}^+_x, \mathcal{R}^-_x)$, where we define $\mathcal{R}^+_x(s, s') = R(s, s') \geq x$ and define $\mathcal{R}^-_x(s, s') = \neg(\neg(R(s, s')) \geq x)$. In interpreting a formula over such a structure, we modify the $\mu L$ semantics as follows:

$$\|M_x, \Box\, \phi\|_{\mathcal{V}} = \lambda s. \bigwedge\{\neg \mathcal{R}^-(s, s') \vee \|M, \phi\|_{\mathcal{V}}(s') \mid \text{all } s'\}$$
$$\|M_x, \Diamond\, \phi\|_{\mathcal{V}} = \lambda s. \bigvee\{\mathcal{R}^+(s, s') \wedge \|M, \phi\|_{\mathcal{V}}(s') \mid \text{all } s'\}$$

Our reduction lemma (Lemma 1) also holds for this extended reduction.

**Lemma 2.** *Let $M$ be a Kripke structure with multi-valued transitions over a finite distributive lattice $L$, with $s$ in $S$, and $x$ in $\mathcal{J}(L)$. Then, letting $M_x$ be the result of the extended reduction, $(M_x, s) \models \phi \Leftrightarrow x \leq [(M, s), \phi]$.*

In the second step, we translate the structure $M_x = (S, s_0, \Theta, \mathcal{R}^+, \mathcal{R}^-)$ to a standard Kripke structure $M'_x = (S', s'_0, \Theta', \mathcal{R}')$. The set of propositions over which $\Theta'$ is defined is $P \cup \{p^+\}$, and

$$\begin{aligned} S' &= \{(s, sign) \mid s \in S, sign \in \{+, -\}\} \\ s'_0 &= (s_0, +) \\ \Theta'(s, sign)(p) &= \text{if } p \equiv p^+ \text{ then } (sign = +) \text{ else } \Theta(s, p) \\ \mathcal{R}'((s, sign), (s', sign')) &= (s, s') \in \mathcal{R}^{sign'}(s, s') \end{aligned}$$

For every state $s$ in $M_x$ there are states $(s, +)$ and $(s, -)$ in $M'_x$. Moreover, every pair $(s, +)$, $(s, -)$ of states in $M'_x$ is strongly bisimilar. Since strong bisimulation preserves $\mu L$ formulas [22], we have that $(s, +)$ satisfies $\phi$ iff $(s, -)$ does.

We also define a translation $T$ that maps formulas of $\mu L$ to formulas of $\mu L$. The translation maps all operators $\oplus$ homomorphically (i.e., $T(\phi_1 \oplus \phi_2) = T(\phi_1) \oplus T(\phi_2)$), except the modal operators. In these cases we have $T(\Box\, \phi) = \Box(p^+ \vee T(\phi))$ and $T(\Diamond\, \phi) = \Diamond(p^+ \wedge T(\phi))$. The correctness condition for the second step is that a formula holds of $M_x$ iff the translated formula holds of $M'_x$.

**Proposition 3.** *Let $M_x$ be a Kripke structure with two transition relations, $M'_x$ be the standard Kripke structure obtained by translation from $M_x$, $s$ be a state of $M_x$, and $\phi$ be a formula of $\mu L$. Then $(M_x, s) \models \phi \Leftrightarrow (M'_x, (s, +)) \models T(\phi)$.*

### 3.3 Related Work

In [2] a reduction is given for three-valued model checking. In [19], reductions are given for total orders, binary products of total orders, and the lattice $2 \times 2 + 2$, which can be obtained from the right-most lattice of Fig. 1 by adding a new top element $f$ above element $a$.

A method [17] with the same generality as ours was discovered independently (see [5]). In the method of [17] each $\mu L$ formula is translated first to a set of formulas in a logic designed specifically for the reduction, then each formula in this set is translated to a $\mu L$ formula. Our approach uses fewer steps, no additional logic, and has simpler proofs (due to the use of Birkhoff's theorem).

In [14], Fitting shows how a many-valued Kripke structure can be transformed to a "multiple-expert" structure, that includes a set of *experts* and a binary *dominates* relation over experts. Although the core idea of our method comes from a construction in the proof of Prop. 5.1 of [14], our work differs in several ways. We reduce to standard Kripke structures rather than multi-expert models, we use $\mu L$ rather than propositional modal logic, we use join-irreducible elements rather than proper prime filters, and most importantly, we treat negation parametrically rather than as relative pseudo-complement. The advantage of our approach to negation is generality; the disadvantage is that it increases the size of the model's propositional valuation.

[18] concerns AC-lattices, which are pairs of graph-isomorphic lattices in which the order relation of one is the inverse of the other. Negation in an AC-lattice is captured as two maps, each mapping an element of one lattice to the isomorphic image in the other. AC-lattices can be used for the analysis of conflict between multiple requirements. A notion of expert similar to Fitting's is used.

It is shown, for finite models, that for each of the two "modes" captured by the two lattices in an AC-lattice, the set of views for which a modal mu-calculus formula holds is equal to the set obtained by an interpretation of the formula as a view set. The result differs from ours in that it is based on AC-lattices, in its treatment of negation, and in that it relates view sets rather than lattice elements directly.

## 4 Extended Alternating Automata

The idea behind alternating automata is to describe successor states through boolean expressions built up from states and truth values using conjunction and disjunction. EAA generalize this idea by allowing expressions built up from states and lattice elements using meet and join. A run of an EAA on an input tree is itself a tree, as in alternating automata. However, each node of the run is now labelled with a lattice element.

With alternating automata, one is interested in whether an accepting run exists on an input tree. With EAA, each accepting run has a value (the value at its root), and one is interested in the set of values of all accepting runs. A fundamental question for EAA, and one that is key for the use of EAA in model checking, is whether this set of values has a maximum element. We show below that this is indeed the case.

**Definitions**. Formally, a *tree* $\tau$ is a subset of $\mathbb{N}^*$ such that if $x \cdot c \in \tau$ then $x \in \tau$ and $x \cdot c' \in \tau$ for all $1 \leq c' < c$. The elements of $\tau$ are called its *nodes*, with $\epsilon$ called the *root*. Given a node $x$ of $\tau$, values of the form $x \cdot i$ in $\tau$ are called the *children* or *successors* of $x$. The number of successors of $x$ is called the *degree* of $x$. A node with no successors is called a *leaf*. Given a set $D \subset \mathbb{N}$, a *$D$-tree* is a tree in which the degree of every node is in $D$. A *$\Sigma$-labeled* tree is a pair $(\tau, T)$ in which $\tau$ is a tree and $T : \mathbb{N}^* \to \Sigma$ is a labeling function.

Let $L = (B, \wedge, \vee)$ be a lattice, and let $\mathcal{B}^+(X)$ stand for the set of terms built from elements in a set $X$ using $\wedge$ and $\vee$. A *tree EAA over $L$* is a tuple $A = (\Sigma, D, S, s_0, \rho, F)$, where $\Sigma$ is a nonempty finite alphabet, $S$ is a nonempty finite set of states, $s_0 \in S$ is the initial state, $F$ is an acceptance condition, $D \subset \mathbb{N}$ is a finite set of arities, and $\rho : S \times \Sigma \times D \to \mathcal{B}^+((\mathbb{N} \times S) \cup B)$ is a transition function, where $\rho(s, a, k) \in \mathcal{B}^+((\{1, \ldots, k\} \times S) \cup B)$ is defined for each $s$ in $S$, $a$ in $\Sigma$, and $k$ in $D$. Various types of acceptance conditions $F$ can be used with EAA, just as in alternating automata, and are discussed below.

A *$v$-run* of a tree EAA $A$ on a $\Sigma$-labeled leafless $D$-tree $(\tau, T)$ is an $\mathbb{N}^* \times S \times B$-labeled tree $(\tau_\sigma, T_\sigma)$. A node in $\tau_\sigma$ labeled by $(x, s, v)$ describes a copy of automaton $A$ that reads the node $x$ of $\tau$ in the state $s$ of $A$ and has value $v \in B$ associated with it. Formally, a $v$-run $(\tau_\sigma, T_\sigma)$ is an $\mathbb{N}^* \times S \times B$-labeled tree, defined as follows.

- $T_\sigma(\epsilon) = (\epsilon, s_0, v)$
- Let $y \in \tau_\sigma$, $T_\sigma(y) = (x, s, v')$, $arity(x) = k$, and $\rho(s, T(x), k) = \theta$. Then there is a (possibly empty) set $Q = \{(c_1, s_1, v_1), \ldots, (c_n, s_n, v_n)\} \subseteq \{1, \ldots, k\} \times S \times B$ such that

- for all $1 \leq i, j \leq n$, $c_i = c_j$ and $s_i = s_j$ implies $v_i = v_j$,
- $Eval(Q, \theta) = v'$, and
- for all $1 \leq i \leq n$, we have $y \cdot i \in \tau_\sigma$ and $T_\sigma(y \cdot i) = (x \cdot c_i, s_i, v_i)$

$Eval(Q, \theta)$ denotes the value of the expression $\theta$ obtained by replacing each term $(c_i, s_i)$ in $\theta$ by $v_i$ if $(c_i, s_i, v_i) \in Q$ or by $\perp$ otherwise.

A $v$-run $\sigma$ is *accepting* if (1) the value associated with each node of the run is not $\perp$ and (2) all infinite branches of the run satisfy the acceptance condition $F$. As with traditional alternating automata, various types of acceptance conditions can be used. For instance, a path $w$ satisfies a *parity acceptance condition* $F = \{F_1, F_2, \ldots, F_n\}$ with $F_1 \subseteq F_2 \subseteq \ldots \subseteq F_n$ if the minimal index $i$ for which some state $s$ in $F_i$ appears infinitely often along $w$ is even. Note that an accepting run can have finite branches: if, for some $y \in \tau_\sigma$, $T_\sigma(y) = (x, s, v)$ and $\rho(s, T(x), arity(x)) = v$ with $v$ in $B$ and $v \neq \perp$, then $y$ does not need to have any successor.

A tree EAA $A$ accepts a $\Sigma$-labeled leafless $D$-tree $(\tau, T)$ with value $v$ if there exists an accepting $v$-run of $A$ on that tree. We define the language $\mathcal{L}_v(A)$ as follows (for $v \neq \perp$): $\mathcal{L}_v(A) = \{(\tau, T) \mid A \text{ accepts } (\tau, T) \text{ with value } v\}$. For convenience, we define $\mathcal{L}_\perp(A)$ as $\{(\tau, T) \mid A \text{ has no accepting run on } (\tau, T)\}$. When $D$ is a singleton, $A$ runs over trees with a fixed branching degree. In particular, a *word EAA* is simply a tree EAA in which $D = \{1\}$.

**Existence of Maximum Value**. We now establish a new, fundamental property of EAA: for any EAA and any input tree, there always exists a maximum value $v$ of $L$ for which the EAA has an accepting $v$-run on the input tree. Note that this property is non-trivial since it is not generally true that, if an EAA has an accepting $v_1$-run and an accepting $v_2$-run on an input tree, then the EAA has an accepting $(v_1 \vee v_2)$-run on this input tree.

**Theorem 3 (Maximum-value theorem).** *Let $A$ be a (finite) tree EAA over a lattice $L$, and let $(\tau, T)$ be a $\Sigma$-labeled leafless $D$-tree. Then the subset $\{v \mid (\tau, T) \in \mathcal{L}_v(A)\}$ of $L$ has a maximum value, which we denote by $Max(A, (\tau, T))$.*

We will write simply $Max(A)$ when $A$ is a word EAA on a 1-letter alphabet.

## 5 Model Checking with EAA

Our model-checking procedure for multi-valued logics using EAA generalizes the automata-theoretic approach to 2-valued model checking with AAs [21]. Our procedure computes the value $[(M, s), \phi]$ defined by a $\mu L$ formula $\phi$ evaluated in state $s$ of a Kripke structure $M$ over a DeMorgan lattice $L$. (Multi-valued transitions in $M$ can be transformed first as discussed in Section 3.2.) In the first step of the procedure we translate $\phi$ to an EAA $A_\phi$. Then we build a product automaton from $A_\phi$ and $M$ in such a way that the maximum value that labels an accepting run of the product automaton is $[(M, s), \phi]$. We now present these steps in detail.

We begin with a translation of $\mu L$ formulas to EAA. The translation is similar to the translation from $\mu L$ to parity alternating automata given in [21] except for the case of atomic propositions, which are mapped to lattice elements in our context. The property we want of the translation is that the value of the maximum accepting run of the EAA for formula $\phi$ and an input tree $(\tau, T)$ agrees with the value $[(\tau, T), \phi]$ defined by the semantics of $\mu L$ (with $(\tau, T)$ viewed as a Kripke structure over $L$).

**Theorem 4.** *Let $\phi$ be a closed $\mu L$ formula and $L$ be a DeMorgan lattice. Then a parity EAA $A_{D,\phi}$ for $\phi$ can be constructed in linear time such that $[((\tau, T), \epsilon), \phi)] = Max(A_{D,\phi}, (\tau, T))$ for every leafless $D$-tree $(\tau, T)$ on $L$.*

In the next step of the procedure, we compute the product of a Kripke structure and an EAA representing a $\mu L$ formula. The product construction defined here is again nearly identical to that given for alternating automata in [21].

**Definition 2.** *Let $\phi$ be a closed $\mu L$ formula, $L$ be a DeMorgan lattice, $M = (S, s_0, \Theta, \mathcal{R})$ be a finite Kripke structure over $L$, with degrees in $D$, and $A_{D,\phi} = (P \to L, D, Q_\phi, q_0, \rho_\phi, F)$ be a parity EAA representing $\phi$. Then the* product automaton *$A_{M,\phi} = (\{a\}, S \times Q_\phi, (s_0, q_0), \rho, F)$ of $M$ and $A_{D,\phi}$ is a parity word EAA over a 1-letter alphabet with at most $O(|S| \cdot |Q_\phi|)$ states, where $\rho$ and $F$ are defined as follows:*

- *For all $q \in Q_\phi$, $s \in S$, if $succ_{\mathcal{R}}(s) = (s_1, \ldots, s_n)$ and $\rho_\phi(q, \Theta(s), n) = \theta$, then $\rho((s, q), a) = \theta'$ where $\theta'$ is obtained from $\theta$ by replacing each atom $(c, q')$ in $\theta$ by $(s_c, q')$.*
- *If $F_\phi = \{F_1, F_2, \ldots, F_m\}$ is a parity acceptance condition, then so is $F = \{(S \times F_1), (S \times F_2), \ldots, (S \times F_m)\}$.*

The product automaton $A_{M,\phi}$ is used to prove the following.

**Theorem 5.** *Let $\phi$ be a closed $\mu L$ formula, $M$ be a finite Kripke structure over a DeMorgan lattice $L$, and $s$ be a state of $M$. Then there exists a parity word EAA $A_{M,\phi}$ over a 1-letter alphabet such that $[(M, s), \phi)] = Max(A_{M,\phi})$.*

In the final step of the procedure, we compute the value $Max(A_{M,\phi})$ of the product EAA.

**Theorem 6.** *Given a parity word EAA $A_{M,\phi}$ over $L$ with a 1-letter alphabet, computing $Max(A_{M,\phi})$ has the same complexity as checking whether the language accepted by a parity word AA with a 1-letter alphabet is nonempty, i.e., can be done in nondeterministic polynomial time.*

Algorithms for computing $Max(A)$ of a word EAA $A$ over a 1-letter alphabet are similar to algorithms for checking emptiness of AAs over a 1-letter alphabet except that the algorithms dealing with EAA propagates values in $L$ instead of values in $\{true, false\}$. The number of iterations for each state can be bounded by $O(|h(L)|)$ where $h(L)$ is the height of $L$ (e.g., [15]). The traditional $\mu L$ model-checking problem is in NP$\cap$co-NP, and this upper bound carries over to the multi-valued case. However, computing $Max(A_{M,\phi})$ can be done more efficiently for some subclasses of $\mu L$. For instance, the EAA for a CTL formula $\phi$ is *weak* [21], and computing the value $Max(A_{M,\phi})$ of the product of a weak EAA with a Kripke structure $M$ can be done in time linear in $|M|$ and $|\phi|$ [3].

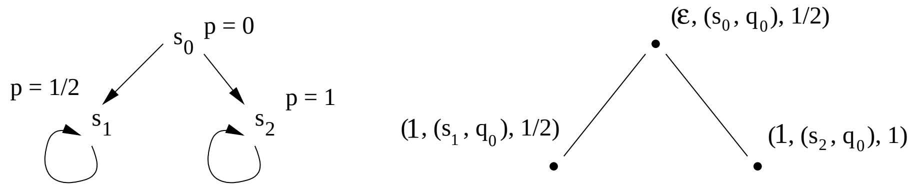


**Fig. 2.** Example Kripke structure $M$ and accepting run

*Example 1.* Consider the $\mu L$ formula $\mu X.p \vee \Box X$, which is equivalent to the CTL formula $AFp$. By translating this formula into an EAA satisfying Theorem 4, we obtain a tree EAA with a single state $q_0$, an acceptance condition $F = \emptyset$ and the following transition function: $\rho(q_0, \sigma, k) = \sigma(p) \vee \bigwedge_{c=1}^{k}(c, q_0)$. We next take the product of this automaton with the Kripke structure $M$ over $L_3$ shown on the left of Figure 2. The figure shows the value of the atomic proposition $p$ at each state. Using the product construction of Definition 2, we obtain a (weak) word EAA over a 1-letter alphabet with no accepting states and the following transition function: $\rho((s_0, q_0), a, 1) = 0 \vee ((s_1, q_0) \wedge (s_2, q_0))$, $\rho((s_1, q_0), a, 1) = 1/2 \vee (s_1, q_0)$, and $\rho((s_2, q_0), a, 1) = 1 \vee (s_2, q_0)$. This EAA has the accepting 1/2-run shown on the right in Figure 2. The value 1/2 is the greatest value $v$ for which there is an accepting $v$-run, so by Theorem 5, we have $[(M, s_0), \mu X.p \vee \Box X] = 1/2$.

## 6 Discussion

As mentioned in the introduction, an advantage of the reduction approach to multi-valued model checking is that it can be implemented using existing model checkers. On the other hand, the direct approach can work in a more "on-the-fly" fashion, computing whatever information is necessary to solve the problem at hand on a demand-driven basis. Indeed, in the reduction approach, only the lattice and Kripke structure are used in building the two-valued Kripke structures, each of which can then be model checked possibly on-the-fly, thus using the formula to guide the verification needs. In contrast, the direct approach can make use of all three inputs together to further limit computational resources. For instance, consider a lattice of $n$ incomparable elements plus a top and bottom element, and suppose the formula we wish to model check is simply the atomic proposition $p$. In the reduction approach we must then perform $n$ model checks. In the direct approach we will perform a single model check that examines only the initial state of the multi-valued Kripke structure and reads only the value of $p$, which requires reading only $log(n)$ bits.

Note that, in a finite-state Kripke structure with finitely-many atomic propositions, at most finitely-many lattice elements will appear. From these, by closing under meet and join, one obtains a finite sublattice of the original lattice. This finite sublattice can be used in place of the original one for multi-valued model checking, with either approach, and thus the size of the original lattice does not matter (and could even be infinite). Finally note that, unlike the reduction approach, the direct approach does not require the lattice to be distributive.

**Acknowledgements.** We thank the anonymous reviewers for their helpful comments. This work was funded in part by NSF CCR-0341658.

## References


1. L. Bolc and P. Borowik. *Many-Valued Logics*. Springer Verlag, 1992.
2. G. Bruns and P. Godefroid. Generalized Model Checking: Reasoning about Partial State Spaces. In *Proc. of CONCUR 2000, LNCS 1877*. Springer-Verlag, 2000.
3. G. Bruns and P. Godefroid. Temporal Logic Query Checking. In *Proc. of LICS '01*, pages 409–417. IEEE, 2001.
4. G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Proc. of CAV '99, LNCS 1633*. Springer-Verlag, 1999.
5. G. Bruns and P. Godefroid. Model checking with multi-valued logics. Technical Report BL03.00018, Bell Labs, Lucent Technologies, May 2003.
6. W. Chan. Temporal-logic queries. In *Proc. of CAV 2000, LNCS 1855*, pages 450–463. Springer-Verlag, 2000.
7. M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel. Multi-valued symbolic model checking. Tech. Report 448, Comp. Sys. Res. Group, Univ. of Toronto, 2001.
8. M. Chechik and W. Easterbrook. A framework for multi-valued reasoning over inconsistent viewpoints. In *Proc. of ICSE '01*, 2001.
9. M. Chechik, W. Easterbrook, and A. Gurfinkel. Model exploration with temporal logic query checking. In *Proc. of FSE '02*, ACM, 2002.
10. M. Chechik, B. Devereux, and A. Gurfinkel. Model-checking infinite state-space systems with fine-grained abstractions using SPIN. In *Proc. of SPIN Workshop on Model-Checking Software*, 2001.
11. B.A. Davey and H.A. Priestly. *Introduction to Lattices and Order*. Cambridge University Press, 1990.
12. E. A. Emerson. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science*, pages 995–1072. Elsevier, 1990.
13. M. Fitting. Many-valued modal logics I. *Fund. Informaticae*, 15:235–254, 1992.
14. M. Fitting. Many-valued modal logics II. *Fund. Informaticae*, 17:55–73, 1992.
15. Ch. Fecht and H. Seidl. A Faster Solver for General Systems of Equations. *Sci. Comp. Programming*, 35(2):137–161, 1999.
16. P. Godefroid and R. Jagadeesan. On the Expressiveness of 3-Valued Models. In *Proc. of VMCAI 2003, LNCS 2575*, pages 206–222. Springer-Verlag, 2003.
17. A. Gurfinkel and M. Chechik. Multi-valued model checking via classical model checking. In *Proc. of CONCUR 2003, LNCS 2761*. Springer-Verlag, 2003.
18. M. Huth and S. Pradhan. Lifting assertion and consistency checkers from single to multiple viewpoints. Technical report 2002/11, Dept. of Computing, Imperial College, London, 2002.
19. B. Konikowska and W. Penczek. Reducing model checking from multi-valued CTL* to CTL*. In *Proc. of CONCUR '02, LNCS 2421*. Springer-Verlag, 2002.
20. D. Kozen. Results on the Propositional Mu-Calculus. *Theoretical Computer Science*, 27:333–354, 1983.
21. O. Kupferman, M. Y. Vardi, and P. Wolper. An Automata-Theoretic Approach to Branching-Time Model Checking. *JACM*, 47(2):312–360, March 2000.
22. C. Stirling. Modal and temporal logics for processes. Notes for Summer School in Logic Methods in Concurrency, C.S. Dept., Åarhus University, Denmark, 1993.
23. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. of Maths*, 5:285–309, 1955.