Generating Counterexamples for Multi-Valued Model-Checking

Arie Gurfinkel and Marsha Chechik

Department of Computer Science, University of Toronto
Toronto ON M5S 3G4, Canada
{arie,chechik}@cs.toronto.edu

Abstract. Counterexamples explain why a desired temporal logic property fails to hold, and as such are considered to be the most useful form of output from model-checkers. Multi-valued model-checking, introduced in [4] is an extension of classical model-checking. Instead of classical logic, it operates on elements of a given De Morgan algebra, e.g. the Kleene algebra [13]. Multi-valued model-checking has been used in a number of applications, primarily when reasoning about partial [2] and inconsistent [10] systems. In this paper we show how to generate counterexamples for multi-valued model-checking. We describe the proof system for a multi-valued variant of CTL, discuss how to use it to generate counterexamples. The techniques presented in this paper have been implemented as part of our symbolic multi-valued model-checker χChek [3].

Keyword: model-checking, De Morgan algebras, counterexamples, witnesses, CTL.

Submission Category: Theoretical paper.

1 Introduction

A classical model-checker can tell the user not only whether a desired temporal property is violated, but also generate a counterexample, explaining the reasons behind the answer. Typically, counterexamples are fairly small and are given in terms of states and transitions of the model; thus, they are readily understood by engineers and can be effectively used for debugging the model. The counterexample generation ability has been one of the major advantages of model-checking in comparison with other verification methods.

Counterexamples are a form of a mathematical proof: to disprove that some property φ holds on all elements of some set S, it is sufficient to produce a single element $s \in S$ such that $\neg\varphi$ holds on s. For model-checking, counterexamples are restricted to universally-quantified formulas and can be viewed as infinite or finite trees starting from the initial state that illustrate failure of a given property [8]. A dual problem is that of computing witnesses to existential properties. A witness is a part of the model that is sufficient to prove that $[\![\varphi]\!](s) = \mathsf{true}$, where φ is a temporal property and s is a state of the model.

In this paper, we study the generation of witnesses and counterexamples for multi-valued model-checking. Multi-valued model-checking, introduced in [4], is an extension of classical model-checking. A multi-valued model-checker operates on elements of a De Morgan (also known as quasi-boolean) algebra $(\mathcal{L}, \sqsubseteq, \neg)$ [6], where $(\mathcal{L}, \sqsubseteq)$ is a finite distributive lattice, $\sqcap$ and $\sqcup$ are meet and join of this lattice, respectively (i.e., $a \sqsubseteq b \equiv a \sqcap b = a$ and $a \sqcup b = b$), and $\neg$ preserves De Morgan laws and involution ($\neg\neg a = a$ for every $a \in \mathcal{L}$). Some examples include the classical logic (T = $\top$ and F = $\bot$ of the lattice, respectively) and Kleene algebra [13] with values T=$\top$, M, F=$\bot$ forming a total order ($\top \geq M \geq \bot$) with negation defined as $\neg$T = F, $\neg$F = T, $\neg$ M = M. Multi-valued models, called χKripke structures, are extensions of classical Kripke structures, where values of variables and the transition relation are multi-valued. Temporal properties are expressed in χCTL — a branching-time temporal logic with the same syntax as CTL but where atomic propositions can evaluate to any $\ell \in \mathcal{L}$. Multi-valued model-checking has been used in a number of applications, primarily when reasoning about partial [2] and inconsistent [10] systems.

Given a temporal property φ, a χKripke structure K, and a De Morgan algebra $\mathcal{L}$, a multi-valued model-checker (such as our model-checker χChek [3]) determines $[\![\varphi]\!]$ — a function mapping each state s of K to the value of φ on it. Thus, we are interested in giving the user an explanation why $[\![\varphi]\!](s) = \ell$, for $\ell \in \mathcal{L}$. To do so, we must explain both $[\![\varphi]\!](s) \sqsupseteq \ell$ and $[\![\varphi]\!](s) \sqsubseteq \ell$. When ℓ is $\top$, $[\![\varphi]\!](s) \sqsubseteq \top$ follows directly from the definition of $\top$ for any formula φ and thus requires no further explanation. Similarly, when ℓ is $\bot$, $[\![\varphi]\!](s) \sqsupseteq \bot$ requires no explanation. Thus, in classical model-checking, only one of the two explanations is given. We keep the same terminology in the multi-valued case, referring to $[\![\varphi]\!] \sqsupseteq \ell$ and $[\![\varphi]\!] \sqsubseteq \ell$ as witnesses and counterexamples, respectively.

In this paper, we show how to automatically generate witnesses and counterexamples for χCTL. To do so, we describe witness and counterexample generation for χECTL (an existential fragment of χCTL) and then give a treatment of negation. Often, counterexamples for existential properties are too large to be feasible. Yet, a partial exploration of such a counterexample may yield useful information. More importantly, this allows us to create a unified framework for giving witnesses and counterexamples for *arbitrary* χCTL properties.

Note that our approach is quite different from the one used by Clarke et al. for classical model-checking [8]. Instead of developing an algorithm to construct witnesses and counterexamples from the model, we first develop a proof system for χECTL, show how to use it to automatically generate proofs, and finally how to extract witnesses and counterexamples from these proofs. Finally, we extend it to full χCTL. A similar approach for classical model-checking was taken in [12]. In this paper, we do not consider proof generation from model-checking with fairness. For a treatment of this issue, please refer to [11]. The rest of this paper is organized as follows: Section 2 reviews the concept of multi-valued model-checking. Our approach to generating proofs for χCTL is introduced in Section 3. We discuss how to use proof rules to construct witnesses and counterexamples in Section 4. The paper is concluded in Section 5 with discussion of related work and the outline of future research directions.

2 Background

In this section, we briefly review fundamentals of classical and multi-valued model-checking and fix some notation.

2.1 CTL Model-Checking

CTL model-checking [7] is an automated technique for verifying properties expressed in a propositional branching-time temporal logic called *Computation Tree Logic* (CTL). A *model* is a Kripke structure $K = (S, R, s_0, A, I)$ where S is a set of states; $R \subseteq S \times S$ is a (total) transition relation; $s_0 \in S$ is an initial state; A is a set of atomic propositions; and $I : S \times A \to \{\top, \bot\}$ is a (total) labeling function. Properties are evaluated on a tree of infinite computations produced by K.

The syntax of CTL is as follows. All atomic propositions are CTL formulas. In addition, if φ and ψ are CTL formulas, then so are $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$. Temporal operators of CTL are: $EX\varphi$ ($AX\varphi$) — φ holds in one (all) of the next states; $EG\varphi$ ($AG\varphi$) — there is a path (for all paths), starting at the current state, where φ holds in every state; $E[\varphi\ U\ \psi](A[\varphi\ U\ \psi])$ — there is a path (for all paths) along which ψ eventually holds, and until that point, φ holds in every state; and $EF\varphi$ ($AF\varphi$) — there is a path (for all paths), along which φ eventually holds. We write $[\![\varphi]\!]^K(s)$ to indicate the value of φ in state s of K. If K is clear from the context, it is omitted from the notation. If a formula φ holds in the initial state, i.e. $[\![\varphi]\!](s_0) = \top$, it is considered to hold in the model.

EX, EG and EU form an adequate set for CTL [7]. These operators are defined as follows:

$$\begin{aligned}
[\![EX\varphi]\!](s) &\triangleq \exists t \in S \cdot R(s,t) \wedge [\![\varphi]\!](t) \\
[\![EG\varphi]\!](s) &\triangleq [\![\nu Z \cdot \varphi\ \wedge\ EXZ]\!](s) \\
[\![E[\varphi\ U\ \psi]]\!](s) &\triangleq [\![\mu Z \cdot \psi \vee (\varphi\ \wedge\ EXZ)]\!](s)
\end{aligned}$$

where $\mu Z \cdot f(Z)$ and $\nu Z \cdot f(Z)$ are the least and the greatest fixpoints of a function f, respectively.

The rest of the operators are defined using EX, EG and EU as follows:

$$\begin{aligned}
[\![AX\varphi]\!](s) &\triangleq \neg[\![EX(\neg\varphi)]\!](s) \\
[\![EF\varphi]\!](s) &\triangleq [\![E[\top\ U\ \varphi]]\!](s) \\
[\![AG\varphi]\!](s) &\triangleq \neg[\![EF(\neg\varphi)]\!](s) \\
[\![AF\varphi]\!](s) &\triangleq \neg[\![EG(\neg\varphi)]\!](s) \\
[\![A[\varphi\ U\ \psi]]\!](s) &\triangleq \neg[\![E[\neg\psi\ U\ \neg\varphi \wedge \neg\psi]]\!](s)\ \wedge\ \neg[\![EG\neg\psi]\!](s)
\end{aligned}$$

2.2 De Morgan Algebras

Here we give the basics of lattice theory and define De Morgan algebras.

Definition 1. *A* lattice *is a partial order ($\mathcal{L}$, $\sqsubseteq$), where every finite subset $B \subseteq \mathcal{L}$ has a least upper bound (called "join" and written $\sqcup B$) and a greatest lower bound (called "meet" and written $\sqcap B$). $\top$ and $\bot$ are the maximal and the minimal elements of a lattice, respectively.*

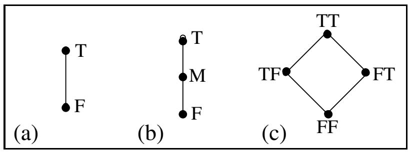

Fig. 1. Examples of a few distributed lattices.

In this paper, if $(\mathcal{L}, \sqsubseteq)$ is a lattice, and the ordering operation $\sqsubseteq$ is clear from context, we refer to it as $\mathcal{L}$.

Definition 2. *A lattice is* distributive *if for all lattice elements* a, b, c*,*

$$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$$
$$a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$$

A few examples of distributive lattices are given in Figure 1.

Definition 1 *Let a lattice $\mathcal{L}$ be given, and let $B \subseteq \mathcal{L}$ and $C \subseteq B$. Then, C is* join-irredundant *in B iff*

1. $\bigsqcup C = \bigsqcup B$, *and*
2. $\forall D \subset C \cdot \bigsqcup D \neq \bigsqcup B$

For example, $\{$TT$\}$ and $\{$TF, FT$\}$ are join-irredundant subsets of a set $\{$TT, TF, FT$\}$ defined over the lattice in Figure 1(c).

We now turn our attention to defining De Morgan algebras.

Definition 3. *A De Morgan algebra is a tuple $(\mathcal{L}, \sqsubseteq, \neg)$, where $(\mathcal{L}, \sqsubseteq)$ is a finite distributive lattice and $\neg$ is any operation that preserves involution ($\neg\neg\ell = \ell$) and De Morgan laws. Conjunction and disjunction are defined using meet and join operations of $(\mathcal{L}, \sqsubseteq)$, respectively.*

In De Morgan algebras, we get $\neg\top = \bot$ and $\neg\bot = \top$, but not necessarily the law of non-contradiction ($\ell \sqcap \neg\ell = \bot$) or excluded middle ($\ell \sqcup \neg\ell = \top$).

We can define several De Morgan algebras using the lattices given in Figure 1. The domain of logical values of the classical logic, referred to as $\mathbf{2}$, is the lattice in Figure 1(a). The three-valued algebra $\mathbf{3}$ (Kleene logic [13]) is defined on the lattice in Figure 1(b), where ¬T = F, ¬F = T, ¬M = M. The four-valued algebra $\mathbf{2x2}$ is defined on the lattice in Figure 1(c), where ¬ TF = FT and ¬ FT = TF. This algebra can be used for reasoning about inconsistency. Note that $\top$ and $\bot$ of the lattice are interpreted as values true and false of the algebra, respectively.

When the negation and the ordering operators of an algebra $(\mathcal{L}, \sqsubseteq, \neg)$ are clear from the context, we refer to it simply as $\mathcal{L}$.

2.3 Multi-Valued Model-Checking

Multi-Valued CTL model-checking [4] is a generalization of the CTL model-checking problem. A multi-valued model-checker receives a De Morgan algebra, a multi-valued

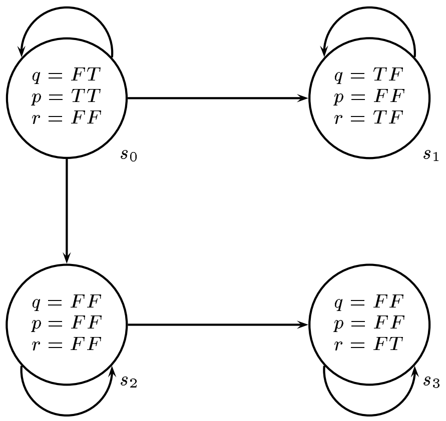

Fig. 2. An example χKripke structure.

model, and a temporal property and determines the value with which this property holds in the model. We define multi-valued models on χ*Kripke structures* — generalizations of Kripke structures, where each atomic proposition and each transition between a pair of states is labeled with values from the algebra. Formally, $K = (S, s_0, \mathbb{R}, I, A, L)$ is a χKripke structure, where $L = (\mathcal{L}, \sqsubseteq, \neg)$ is a De Morgan algebra; A is a (finite) set of atomic propositions; $s_0 \in S$ is the initial state; $\mathbb{R} : S \times S \to \mathcal{L}$ is the multi-valued transition relation; $I : S \times A \to \mathcal{L}$ is a (total) labeling function, such that for each atomic proposition $a \in A$, $I(s, a) = \ell$ means that the variable a has value ℓ in state s. Thus, any Kripke structure is also a χKripke structure over the algebra $\mathbf{2}$.

An example χKripke structure for the algebra $\mathbf{2x2}$ is given in Figure 2. When presenting finite-state machines graphically, we follow the convention of not showing $\bot$ transitions and not labeling $\top$ transitions, to avoid clutter.

Properties are specified in a multiple-valued extension of CTL called χCTL. Given a De Morgan algebra $\mathcal{L}$, χCTL($\mathcal{L}$) has the same syntax as CTL, except that any $\ell \in \mathcal{L}$ is also a χCTL($\mathcal{L}$) formula. The semantics follows:

$$\begin{aligned}
[\![\ell]\!](s) &\triangleq \ell, \text{ for } \ell \in \mathcal{L} \\
[\![a]\!](s) &\triangleq I(s, a), \text{ for } a \in A \\
[\![\neg\varphi]\!](s) &\triangleq \neg[\![\varphi]\!](s) \\
[\![\varphi \wedge \psi]\!](s) &\triangleq [\![\varphi]\!](s) \sqcap [\![\psi]\!](s) \\
[\![\varphi \vee \psi]\!](s) &\triangleq [\![\varphi]\!](s) \sqcup [\![\psi]\!](s) \\
[\![EX\varphi]\!](s) &\triangleq \textstyle\bigsqcup_{t \in S}(\mathbb{R}(s, t) \sqcap [\![\varphi]\!](t))
\end{aligned}$$

The semantics of the EX operator comes from extending the notion of existential quantification for multi-valued reasoning through the use of disjunction [1, 16]. The other operators are defined as their CTL counterparts (see Section 2.1), where $\vee$ and $\wedge$ are interpreted as lattice $\sqcup$ and $\sqcap$, respectively.

We also introduce bounded versions of EU and EG operators.

$$[\![E[\varphi\ U_i\ \psi]]\!](s) \triangleq \begin{cases} [\![\psi]\!](s) & \text{if } i = 0 \\ [\![\psi \vee \varphi \wedge EXE[\varphi\ U_{i-1}\ \psi]]\!](s) & \text{if } i > 0 \end{cases}$$

$$[\![EG_i\varphi]\!](s) \triangleq \begin{cases} [\![\varphi \wedge EX\top]\!](s) & \text{if } i = 0 \\ [\![\varphi \wedge EXEG_{i-1}\varphi]\!](s) & \text{if } i > 0 \end{cases}$$

Intuitively, the bound i corresponds to the restriction of the operators to finite paths of length at most i. Formally, they correspond to the ith approximation in their respective fixpoint computation. χCTL($\mathcal{L}$) operators EU and EG are the limits of EU_i and EG_i, respectively:

$$[\![E[\varphi\ U\ \psi]]\!](s) \triangleq [\![E[\varphi\ U_\infty\ \psi]]\!](s)$$
$$[\![EG\varphi]\!](s) \triangleq [\![EG_\infty\varphi]\!](s)$$

Other χCTL operators are computed from EX, EG and EU in the same manner as their CTL counterparts. Note that CTL is the same as χCTL defined over the classical logic $\mathbf{2}$.

Given an algebra $\mathcal{L}$, we use χACTL($\mathcal{L}$) and χECTL($\mathcal{L}$) to refer to the universal and the existential fragments of χCTL($\mathcal{L}$), respectively. We also use p and q to stand for arbitrary atomic propositions, s and t to represent states, and φ and ψ to represent χCTL formulas. $\{s\}$ expresses a formula that evaluates to $\top$ at state s and $\bot$ otherwise, i.e. $[\![\{s\}]\!](t) \triangleq (s = t)$. $\overline{\{s\}}$ represents the negation of $\{s\}$.

We have implemented a symbolic model-checker χChek [3] that receives a χKripke structure K and a χCTL formula φ and returns an element of the algebra corresponding to the value of φ in K. For example, if the algebra is $\mathbf{2}$, χChek returns true if φ holds in K and false if it does not: this is the classical model-checking. For more information about multi-valued model-checking, please consult [3, 4].

3 Proof Rules for χCTL

In this section, we develop a proof system that allows us to generate proofs for χCTL properties over χKripke structures. We begin by giving proof rules and axioms for reasoning about χKripke structures and De Morgan algebras. We then proceed with proof rules for witnesses and counterexamples for χECTL and then extend our framework to deal with negation and χACTL, resulting in full χCTL.

3.1 Preliminaries

Since we are interested in proving statements about a particular χKripke structure K expressed in a particular De Morgan algebra $\mathcal{L}$, we assume that our proof system incorporates all axioms and proof rules of propositional logic and De Morgan algebras. Several of such proof rules are given in Figure 3. For example, in the one-point rule, f is a predicate and x is some element of D. Intuitively, the one-point rule of propositional logic states that to justify an existential statement $\exists x \in D \cdot f(x)$, one simply

$$\frac{a}{a \vee b}\ \vee\text{-intro} \qquad \frac{a \quad b}{a \wedge b}\ \wedge\text{-intro}$$

$$\frac{b}{a \vee b}\ \vee\text{-intro} \qquad \frac{f(x')}{\exists x \in D \cdot f(x)}\ \text{one-point rule}$$

$$\frac{a \sqsupseteq \ell \qquad b \sqsupseteq \ell}{a \sqcap b \sqsupseteq \ell}\ \sqcap\text{-intro} \qquad \frac{a \sqsubseteq \ell \qquad b \sqsubseteq \ell}{a \sqcup b \sqsubseteq \ell}\ \sqcup\text{-intro}$$

$$\frac{a \sqsubseteq \neg\ell}{\neg a \sqsupseteq \ell}\ \text{De Morgan-not} \qquad \frac{a \sqsupseteq \neg\ell}{\neg a \sqsubseteq \ell}\ \text{De Morgan-not}$$

$$\frac{f(d_1) \qquad f(d_2) \qquad \ldots \qquad f(d_n)}{\forall d \in D \cdot f(d)}\ \text{finite quantification, with } \bigcup_{i=1}^{n}\{d_i\} = D$$

Fig. 3. Some proof rules of propositional logic and De Morgan algebras.

needs to exhibit an element $x' \in D$ for which $f(x')$ holds. Note that in what follows, we only consider quantification over finite domains. Thus, although we do use universal and existential quantification, the proof system remains propositional rather than first order.

In addition, we also assume that the axiomatization of a given algebra $\mathcal{L}$ is available. Such an axiomatization defines relations $\sqsupseteq$ and $\neg$. For example, some of the axioms describing the algebra $\mathbf{2x2}$ shown in Figure 1(c) are:

$$\begin{array}{ll} \text{TT} \sqsupseteq \text{TF} & \text{TF} \sqsupseteq \text{FF} \\ \neg\text{TT} = \text{FF} & \neg\text{TF} = \text{FT} \end{array}$$

In addition, we assume that all axioms of the theory of χKripke structures and the axiomatization of a particular χKripke structure K are also available. Such axioms define the transition relation $\mathbb{R}$ and the state labeling function I. For example, an axiom of χKripke structures is that $I(s, a)$ is defined for any atomic proposition a and any state $s \in S$. Some of the axioms of the χKripke structure in Figure 2, specified using an algebra $\mathbf{2x2}$, are:

$$\begin{array}{ll} \mathbb{R}(s_0, s_1) = \text{TT} & \mathbb{R}(s_0, s_3) = \text{FF} \\ I(s_0, p) = \text{TT} & I(s_0, q) = \text{FT} \end{array}$$

3.2 Proof Rules for Witnesses for χECTL

Our initial goal is to develop a sound and complete proof system that allows us to prove validity of sentences of the form $[\![\varphi]\!](s) \sqsupseteq \ell$, where φ is a χECTL formula, ℓ is a lattice value, and s is a state of a given χKripke structure K. We refer to these as *witnesses*.

The proof rules for non-temporal operators and EX are shown in Figure 4. They follow directly from the definitions of the corresponding operators. For example, the $\wedge$-rule states that in order to prove $[\![\varphi \vee \psi]\!] \sqsupseteq \ell$, we need to find algebra values ℓ_1 and ℓ_2 such that $[\![\varphi]\!](s) \sqsupseteq \ell_1$, $[\![\psi]\!](s) \sqsupseteq \ell_2$, and their join is above ℓ. Thus, this rule introduces

$$\frac{\ell_1 \sqsupseteq \ell}{[\![\ell_1]\!](s) \sqsupseteq \ell} \text{ value-rule} \qquad \frac{\neg\ell_1 \sqsupseteq \ell}{[\![\neg\ell_1]\!](s) \sqsupseteq \ell} \text{ neg-value-rule}$$

$$\frac{\exists \ell_1 \in \mathcal{L} \cdot (I(s,p) = \ell_1) \ \wedge \ (\ell_1 \sqsupseteq \ell)}{[\![p]\!](s) \sqsupseteq \ell} \text{ atomic-rule} \quad \frac{\exists \ell_1 \in \mathcal{L} \cdot (\neg I(s,p) = \ell_1) \ \wedge \ (\ell_1 \sqsupseteq \ell)}{[\![\neg p]\!](s) \sqsupseteq \ell} \text{ neg-atomic-rule}$$

$$\frac{[\![\varphi]\!](s) \sqsupseteq \ell \ \wedge \ [\![\psi]\!](s) \sqsupseteq \ell}{[\![\varphi \wedge \psi]\!](s) \sqsupseteq \ell} \ \wedge\text{-rule} \qquad \frac{\begin{array}{c}\exists \ell_1, \ell_2 \in \mathcal{L} \cdot ([\![\varphi]\!](s) \sqsupseteq \ell_1) \ \wedge \\ ([\![\psi]\!](s) \sqsupseteq \ell_2) \ \wedge \ (\ell_1 \sqcup \ell_2 \sqsupseteq \ell)\end{array}}{[\![\varphi \vee \psi]\!](s) \sqsupseteq \ell} \ \vee\text{-rule}$$

$$\frac{\begin{array}{c}\exists t_1, \ldots, t_n \in S \cdot \exists \ell_1, \ldots, \ell_n \in \mathcal{L} \cdot ([\![\mathbb{R}(s,t_1) \wedge \varphi]\!](t_1) \sqsupseteq \ell_1) \\ \wedge \ \cdots \ \wedge \ ([\![\mathbb{R}(s,t_n) \wedge \varphi]\!](t_n) \sqsupseteq \ell_n) \ \wedge \ (\bigsqcup_{i=1}^{n} \ell_i) \sqsupseteq \ell\end{array}}{[\![EX\varphi]\!](s) \sqsupseteq \ell} \ EX$$

Fig. 4. Proof rules for non-temporal operators and EX.

$$\frac{[\![\psi]\!](s) \sqsupseteq \ell}{[\![E[\varphi U_0 \psi]]\!](s) \sqsupseteq \ell} \ EU_0 \quad \frac{[\![\psi \vee \varphi \wedge EXE[\varphi U_{n-1} \psi]]\!](s) \sqsupseteq \ell}{[\![E[\varphi U_n \psi]]\!](s) \sqsupseteq \ell} \ EU_i$$

$$\frac{\exists n \in nat \cdot [\![E[\varphi U_n \psi]]\!](s) \sqsupseteq \ell}{[\![E[\varphi U \psi]]\!](s) \sqsupseteq \ell} \ EU$$

$$\frac{[\![\varphi \wedge EXE[\varphi U \varphi \wedge \{s\}] \vee \varphi \wedge EXEG(\varphi \wedge \overline{\{s\}})]\!](s) \sqsupseteq \ell}{[\![EG\varphi]\!](s) \sqsupseteq \ell} \ EG$$

Fig. 5. Proof rules for EU and EG.

two existential quantifiers, which are typically eliminated by several applications of the one-point rule shown in Figure 3.

The proof rules for the bounded EU are given in Figure 5 and follow directly from the definition of this operator. To derive the rule for the unbounded EU, we start by noting the monotonicity of EU_i:

Proposition 1. *Let φ, ψ be ECTL formulas and $i, j \in nat$. Then,*

$$i \geq j \Rightarrow \forall s \in S \cdot ([\![E[\varphi \ U_i \ \psi]]\!](s) \sqsupseteq [\![E[\varphi \ U_j \ \psi]]\!](s))$$

The proof rule for unbounded EU (given in Figure 5) is obtained by combining Proposition 1 with the fact that the unbounded EU is an upper bound of the bounded EU_i:

$$\forall i \in nat \cdot [\![E[\varphi \ U \ \psi]]\!] \sqsupseteq [\![E[\varphi \ U_i \ \psi]]\!]$$

Note that since we assume that the state space is finite, the EU rule is actually bidirectional. That is, for a given Kripke structure K, there always exists a natural number n, which depends on the diameter of the directed graph induced by K, such that $E[\varphi \ U \ \psi] = E[\varphi \ U_n \ \psi]$.

To complete our proof system, we still need to find a proof rule for EG. Unfortunately, we cannot proceed as in the previous cases and use the χECTL equivalence

$EG\varphi = \varphi \wedge EXEG\varphi$ to define the proof rule. Doing so would result in a proof system which is not complete, since this proof rule can be potentially applied an infinite number of times.

Instead, note that $[\![EG\varphi]\!](s)$ is the result of evaluating $G\varphi$ on all infinite paths emanating from the state s. Moreover, since we are dealing with finite state systems, every infinite path can be decomposed into a finite (possibly empty) prefix and a finite repeating suffix. Thus, we can decompose $[\![EG\varphi]\!](s)$ into the join of the EG restricted to all non-trivial cycles around s, and EG restricted to all infinite paths that do not contain s in the future.

First, we consider the restriction of $[\![EG\varphi]\!](s)$ to all non-trivial cycles around s. The set of non-trivial cycles around s is exactly the set of paths along which s occurs infinitely often[1]. Furthermore, since our starting state is s, any infinite path along which s occurs infinitely often is equivalent to a finite path from s to itself. Thus, to evaluate $[\![EG\varphi]\!](s)$ restricted to cycles around s, it is sufficient to consider only finite paths from s to s. This intuition is formalized in the following theorem, the proof of which is available in [11]:

Theorem 1. *Let φ be an ECTL formula and s be a state of a Kripke structure. Then,*

$$[\![EG\varphi]\!](s) = [\![(\varphi \wedge EXE[\varphi\ U\ \varphi \wedge \{s\}]) \vee (\varphi \wedge EXEG(\varphi \wedge \overline{\{s\}}))]\!](s)$$

A proof rule for a EG witness is given in Figure 5.

Theorem 2. *The proof system for witnesses for χECTL is sound and complete.*

For the proof of this theorem, see [11].

3.3 Automatic Proof Generation for Witnesses for χECTL

Given a statement $[\![\varphi]\!](s) \sqsupseteq \ell$, we are interested in an automated proof of its validity. We can achieve this by embedding the proof system of Section 3.2 into an automated theorem prover, such as PVS [15], and using its facilities for proof generation. This is always possible, because we assume that the state space of χKripke structures is finite. We can also use a multi-valued theorem prover that already includes the encoding of a large class of multi-valued algebras [17]. A more efficient approach is to use the model-checker as a decision procedure for (a) deciding the validity of a given subformula (so that our proof generator avoids exploring irrelevant proof branches) and for (b) applying the one-point rule. We call this decision procedure `modelCheck` and assume that `modelCheck`(φ, s) computes $[\![\varphi]\!](s)$. We also assume the presence of `qblat` — a decision procedure for determining the validity of De Morgan lattice equalities and inequalities.

We start with a statement $[\![p]\!](s) \sqsupseteq \ell$, where p is an atomic proposition, and apply the atomic-rule (see Figure 4) to obtain

$$\exists \ell_1 \in \mathcal{L} \cdot (I(s,p) = \ell_1) \wedge (\ell_1 \sqsupseteq \ell)$$

We now need to decide whether it is possible to apply the one-point rule by instantiating ℓ_1.

[1] This is referred to as *fair-EG*, where the fairness condition is given by a single formula $\{s\}$ [7].

(a)

proc `atomicOnePoint`(p, s, ℓ)
  $k :=$ `modelCheck`(p, s)
  if `qblat`$(k \sqsupseteq \ell))$ **then**
    apply one-point rule substituting k for ℓ_1
  else
    terminate with **invalid**
  end if
end proc

(b)

proc `orOnePoint`(φ, ψ, s, ℓ)
  $k_\varphi :=$ `modelCheck`(φ, s)
  $k_\psi :=$ `modelCheck`(ψ, s)
  if `qblat`$(k_\varphi \sqcup k_\psi \sqsupseteq \ell)$ **then**
    apply one-point rule substituting k_φ for ℓ_1, k_ψ for ℓ_2
  else
    terminate with **invalid**
  end if
end proc

(c)

proc `euOnePoint`(φ, ψ, s, ℓ)
  $i := 0$
  $eu =$ `modelCheck`$(E[\varphi U \psi], s)$
  $eui = \bot$
  while $eui \neq eu$ **and not** `qblat`$(eui \sqsupseteq \ell)$ **do**
    $eui :=$ `modelCheck`$(E[\varphi U_i \psi], s)$
    $i := i + 1$
  end while
  if `qblat`$(eui \sqsupseteq \ell)$ **then**
    apply one-point rule substituting i for n
  else
    terminate with **invalid**
  end if
end proc

(d)

proc `exOnePoint`(φ, s, ℓ)
  $k :=$ `modelCheck`$(EX\varphi, s)$
  if not `qblat`$(k \sqsupseteq \ell)$ **then**
    terminate with **invalid**
  end if
  $(r_1, p_1) \ldots (r_n, p_n) :=$ `exWitness`(φ, s)
  apply one-point rule substituting (r_i, p_i) for (t_i, ℓ_i)
end proc

Fig. 6. Algorithms for automatic proof generation.

Proposition 2. *Let p be an atomic proposition and let s be a state of a χKripke structure K. Then,*

$$I(s, p) = \ell \quad \Leftrightarrow \quad \texttt{modelCheck}(p, s) = \ell$$

Therefore, the one-point rule is applicable if and only if the statement

$$I(s, p) = \texttt{modelCheck}(p, s) \quad \wedge \quad \texttt{modelCheck}(p, s) \sqsupseteq \ell$$

is valid. Since it is a conjunction, to decide its validity we must show that both $I(s, p) =$ `modelCheck`(p, s) and `modelCheck`$(p, s) \sqsupseteq \ell$ are valid. The validity of the first conjunct follows from the definition of `modelCheck`(p, s); therefore, it is sufficient to just establish the validity of `modelCheck`$(p, s) \sqsupseteq \ell$. As the result, we obtain the algorithm given in Figure 6(a). The case of $[\![\neg p]\!](s) \sqsupseteq \ell$ is handled similarly.

Next, we consider the boolean connectives. Given a statement of the form $[\![\varphi \vee \psi]\!](s) \sqsupseteq \ell$, we apply the $\vee$-rule shown in Figure 4. Using the monotonicity of the $\sqcup$ operator, we get the following proposition:

Proposition 3. *Let φ and ψ be χECTL formulas, let s be a state of a χKripke structure, and let ℓ be a lattice element. Then,*

$$\begin{aligned} &\exists \ell_1, \ell_2 \in \mathcal{L} \cdot [\![\varphi]\!](s) \sqsupseteq \ell_1 \wedge [\![\psi]\!](s) \sqsupseteq \ell_2 \wedge \ell_1 \sqcup \ell_2 \sqsupseteq \ell \\ \Leftrightarrow\ &\exists \ell_3, \ell_4 \in \mathcal{L} \cdot [\![\varphi]\!](s) = \ell_3 \wedge [\![\psi]\!](s) = \ell_4 \wedge \ell_3 \sqcup \ell_4 \sqsupseteq \ell \end{aligned}$$

Using the fact that $[\![\varphi]\!](s) = \texttt{modelCheck}(\varphi, s)$, we see that the one-point rule is applicable if and only if instantiating ℓ_1 to $\texttt{modelCheck}(\varphi, s)$ and ℓ_2 to $\texttt{modelCheck}(\psi, s)$ does not result in invalid statements. As in the previous case, this simplifies to requiring that $\ell_1 \sqcup \ell_2 \sqsupseteq \ell$ be valid for the instantiated values. The resulting algorithm is shown in Figure 6(b). The $\wedge$ operator is handled similarly.

We now examine the case of analyzing the unbounded EU operator. Given a statement $[\![E[\varphi\ U\ \psi]]\!](s) \sqsupseteq \ell$, we first apply the EU-rule, shown in Figure 5. The next step is to find an instantiation of n for the one-point rule. Recall that bounded EU_i is monotone when viewed as a function of i (by Proposition 1). Moreover, it is bounded above by the unbounded EU. Therefore, we can find the instantiation of n by a linear search, starting from $n = 0$. The algorithm for the application of the one-point rule is given in Figure 6(c). The intermediate computations performed by this algorithm are exactly those done by a symbolic multi-valued model-checking algorithm. Thus, if the results of the intermediate computations performed by $\texttt{modelCheck}(E[\varphi\ U\ \psi], s)$ are available, a more efficient binary search can replace the linear one.

For example, consider the χKripke structure in the Figure 2 and assume that we want to prove that $[\![E[p\ U\ q]]\!](s_0) \sqsupseteq$ TT. After the application of the EU-rule, we get

$$\exists n \in nat \cdot [\![E[p\ U_n\ q]]\!](s_0) \sqsupseteq \mathrm{TT}$$

To apply the one-point rule, we first try $[\![E[p\ U_0\ q]]\!](s_0) = [\![q]\!](s_0) = \mathrm{FT} \not\sqsupseteq \mathrm{TT}$. Increasing the bound, we get $[\![E[p\ U_1\ q]]\!](s_0) = \mathrm{TT}$, and therefore we can apply the one-point rule by instantiating n to 1.

Finally, given the statement $[\![EX\varphi]\!](s) \sqsupseteq \ell$, we first apply the EX-rule (see Figure 4) and then eliminate the existential quantifiers by applying the one-point rule. For example, suppose we want to prove that $[\![EXq]\!](s_0) \sqsupseteq$ TT in the χKripke structure in Figure 2. First, we apply the EX-rule, obtaining

$$\frac{\begin{array}{c}\exists \ell_1, \ldots, \ell_n \in \mathcal{L} \cdot \exists t_1, \ldots, t_n \in S \cdot ([\![\mathbb{R}(s_0, t_1) \wedge q]\!](t_1) \sqsupseteq \ell_1) \\ \wedge \cdots \wedge ([\![\mathbb{R}(s_0, t_n) \wedge q]\!](t_n) \sqsupseteq \ell_n) \wedge (\bigsqcup_{1 \le i \le n} \ell_i) \sqsupseteq \mathrm{TT}\end{array}}{[\![EXq]\!](s_0) \sqsupseteq \mathrm{TT}}\ EX$$

We now have to apply the one-point rule to instantiate the pairs $\{(\ell_i, t_i) \mid 1 \le i \le n\}$. For notational convenience, we introduce a function *img* : $S \to \mathcal{L}$, defined as

$$img(x) \triangleq [\![\mathbb{R}(s, x) \wedge \varphi]\!](x)$$

Note that $[\![EX\varphi]\!](s) = \bigsqcup_{t \in S} img(t)$.

Proposition 4. *Let $K = (S, s_0, I, \mathbb{R}, A, \mathcal{L})$ be a χKripke structure, $s \in S$ an arbitrary state of K, φ a χCTL($\mathcal{L}$) formula, and* img *as defined above. Then,*

$$\begin{array}{l}\exists \ell_1, \ldots, \ell_n \in \mathcal{L} \cdot \exists t_1, \ldots, t_n \in S \cdot \\ \quad ([\![\mathbb{R}(s, t_1) \wedge \varphi]\!](t_1) \sqsupseteq \ell_1) \wedge \cdots \wedge ([\![\mathbb{R}(s, t_n) \wedge \varphi]\!](t_n) \sqsupseteq \ell_n) \\ \quad \wedge((\bigsqcup_{1 \le i \le n} \ell_i) \sqsupseteq \ell) \\ \Leftrightarrow \\ \quad (\bigsqcup_{t \in S} img(t)) \sqsupseteq \ell\end{array}$$

Thus, in order to apply the one-point rule, we must find a subset U of S such that $(\bigsqcup_{t \in U} \mathit{img}(t)) \sqsupseteq \ell$. If $U = \{u_1, \ldots, u_n\}$ is such a set, we can apply the one point rule by instantiating t_i to u_i, and ℓ_i to $\mathit{img}(u_i)$. Of course, we can always let $U = S$; however, this unnecessarily increases the size of the proof.

Alternatively, we can obtain U by considering the range of the function *img*. Let $U' = \{\ell_i \mid \exists t_i \in S \cdot \mathit{img}(t_i) = \ell_i\}$. Clearly, it is smaller than the size of the state space $|S|$ and the size of the De Morgan algebra $\mathcal{L}$, that is, $|U'| \leq \min\{|S|, |\mathcal{L}|\}$. Furthermore, given $U' = \{u'_1, \ldots, u'_n\}$, we can construct the set U by letting u_i be any element of $\mathit{img}^{-1}(u'_i)$. In our example, we take representatives from the inverse of $\mathit{img}(S) = \{\mathrm{TF}, \mathrm{FT}\}$: $s_0 \in \mathit{img}^{-1}(\mathrm{FT})$, and $s_1 \in \mathit{img}^{-1}(\mathrm{TF})$. Finally, we apply the one-point rule, obtaining

$$[\![\mathbb{R}(s_0, s_0) \wedge q]\!](s_0) \sqsupseteq \mathrm{FT} \;\wedge\; [\![\mathbb{R}(s_0, s_1) \wedge q]\!](s_1) \sqsupseteq \mathrm{TF} \;\wedge\; \mathrm{FT} \sqcup \mathrm{TF} \sqsupseteq \mathrm{TT}$$

Note that in contrast to other χECTL operators, the procedure for generating a proof for $[\![EX\varphi]\!](s) \sqsupseteq \ell$ does not use the model-checker as a simple black-box. In fact, it can only be accomplished efficiently if the model-checker can produce the states comprising the witness.

The algorithm for the application of the one-point rule is given in Figure 6(d). The algorithm makes use of the function $\texttt{exWitness}(\varphi, s)$ that computes the witness for $[\![EX\varphi]\!](s) = \texttt{modelCheck}(EX\varphi, s)$. The witness is returned as a set of pairs (t_i, ℓ_i), such that $\mathit{img}(t_i) = \ell_i$, and $\bigsqcup_{1 \leq i \leq n} \ell_i = \texttt{modelCheck}(EX\varphi, s)$.

3.4 Extending the Proof System to χCTL

Here we extend the results presented earlier in this section to counterexamples for χECTL, and finally to witnesses and counterexamples for full χCTL.

Extension to counterexamples for χECTL Our goal is to extend the proof system to deal with statements of the form $[\![\varphi]\!](s) \sqsubseteq \ell$, where φ is in χECTL. We refer to these as *counterexamples*.

The proof rules for propositional operators are similar to their witness counterparts and are shown in Figure 7. Note that because of the duality of $\wedge$ and $\vee$, the witness proof rules for $\wedge$ and $\vee$ are similar to the counterexample proof rules of $\vee$ and $\wedge$, respectively.

The counterexample proof rule for EX is also shown in Figure 7. To prove that $[\![EX\varphi]\!](s) \sqsubseteq \ell$, we must prove that $[\![\mathbb{R}(s,t) \wedge \varphi]\!](t) \sqsubseteq \ell$ for every state t since EX is defined as a join over all states of the χKripke structure. Thus, even one application of the EX rule may increase the size of the proof tree dramatically.

Finally, the counterexample proof rules for EG and EU are given in Figure 8. Note that the counterexample rule for EG is similar to the witness rule for EU, and is based on the fact that bounded EG_i approximates the unbounded EG from above.

The counterexample proof rule for EU is the most complicated one. It is similar to the witness proof rule for EG in a sense that it decomposes the overall proof into (a) the proof about the current state, and (b) the proof about the rest of the system excluding the current state. It is formally justified by the following theorem.

$$\frac{\ell_1 \sqsubseteq \ell}{[\![\ell_1]\!](s) \sqsubseteq \ell} \text{ value-rule} \qquad \frac{\neg \ell_1 \sqsubseteq \ell}{[\![\neg \ell_1]\!](s) \sqsubseteq \ell} \text{ neg-value-rule}$$

$$\frac{\exists \ell_1 \in \mathcal{L} \cdot (I(s,p) = \ell_1) \ \wedge \ (\ell_1 \sqsubseteq \ell)}{[\![p]\!](s) \sqsubseteq \ell} \text{ atomic-rule} \ \frac{\exists \ell_1 \in \mathcal{L} \cdot (\neg I(s,p) = \ell_1) \ \wedge \ (\ell_1 \sqsubseteq \ell)}{[\![\neg p]\!](s) \sqsubseteq \ell} \text{ neg-atomic-rule}$$

$$\frac{\begin{array}{l}\exists \ell_1, \ell_2 \in \mathcal{L} \cdot ([\![\varphi]\!](s) \sqsubseteq \ell_1) \ \wedge \\ ([\![\psi]\!](s) \sqsubseteq \ell_2) \ \wedge \ (\ell_1 \sqcap \ell_2 \sqsubseteq \ell)\end{array}}{[\![\varphi \wedge \psi]\!](s) \sqsubseteq \ell} \ \wedge\text{-rule} \qquad \frac{[\![\varphi]\!](s) \sqsubseteq \ell \ \wedge \ [\![\psi]\!](s) \sqsubseteq \ell}{[\![\varphi \vee \psi]\!](s) \sqsubseteq \ell} \ \vee\text{-rule}$$

$$\frac{\forall t \in S \cdot [\![\mathbb{R}(s,t) \wedge \varphi]\!](t) \sqsubseteq \ell}{[\![EX\varphi]\!](s) \sqsubseteq \ell} \ EX$$

Fig. 7. Proof rules for counterexamples for non-temporal operators and EX.

$$\frac{[\![\varphi \wedge EX\top]\!](s) \sqsubseteq \ell}{[\![EG_0\varphi]\!](s) \sqsubseteq \ell} \ EG_0 \ \frac{[\![\varphi \wedge EXEG_{n-1}\varphi]\!](s) \sqsubseteq \ell}{[\![EG_n\varphi]\!](s) \sqsubseteq \ell} \ EG_i$$

$$\frac{\exists n \in nat \cdot [\![EG_n\varphi]\!](s) \sqsubseteq \ell}{[\![EG\varphi]\!](s) \sqsubseteq \ell} \ EG$$

$$\frac{[\![\psi \vee \varphi \wedge EXE[\varphi \wedge \overline{\{s\}} \ U \ \psi \wedge \overline{\{s\}}]]\!](s) \sqsubseteq \ell}{[\![E[\varphi \ U \ \psi]]\!](s) \sqsubseteq \ell} \ EU$$

Fig. 8. Proof rules for counterexamples for EG and EU.

Theorem 3. *Let φ, ψ be χCTL formulas, s be a state of a χKripke structure. Then,*

$$[\![E[\varphi \ U \ \psi]]\!](s) = [\![\psi \vee \varphi \wedge EXE[\varphi \wedge \overline{\{s\}} \ U \ \psi \wedge \overline{\{s\}}]]\!](s)$$

Theorem 4. *The proof system for counterexamples for χECTL is sound and complete.*

Extension to χCTL To extend our proof system to witnesses and counterexamples for full χCTL, we must extend the not-rule of De Morgan algebras (see Figure 3) to χCTL, and provide proof rules for χACTL— the universal fragment χCTL. The extension of the not-rule of De Morgan algebras is trivial, and yields the witness and counterexample proof rules shown in Figure 9. The proof rules for χACTL can be obtained from the well-known dualities between χECTL and χACTL (see Section 2.1). For example, the duality $AX\varphi = \neg EX\neg\varphi$ yields the witness and counterexample proof rules for AX shown in Figure 9. The rest of χACTL rules are derived similarly.

Theorem 5. *The proof system for χCTL is sound and complete.*

$$\frac{[\![\varphi]\!](s) \sqsubseteq \neg\ell}{[\![\neg\varphi]\!](s) \sqsupseteq \ell}\ \text{not-rule} \qquad \frac{[\![\varphi]\!](s) \sqsupseteq \neg\ell}{[\![\neg\varphi]\!](s) \sqsubseteq \ell}\ \text{not-rule}$$

$$\frac{[\![\neg EX \neg\varphi]\!](s) \sqsupseteq \ell}{[\![AX\varphi]\!](s) \sqsupseteq \ell}\ AX \quad \frac{[\![\neg EX \neg\varphi]\!](s) \sqsubseteq \ell}{[\![AX\varphi]\!](s) \sqsubseteq \ell}\ AX$$

Fig. 9. Witness (left) and counterexample (right) proof rules for negation and AX.

(a)

$$\cfrac{\cfrac{[\![\mathbb{R}(s,t'_1) \wedge \varphi]\!](t'_1) \sqsupseteq \ell_1 \qquad \cdots \qquad [\![\mathbb{R}(s,t'_n) \wedge \varphi]\!](t'_n) \sqsupseteq \ell_n \qquad \bigsqcup_{1 \le i \le n} \ell'_i \sqsupseteq \ell}{\begin{array}{l}\exists \ell_1, \ldots, \ell_n \in \mathcal{L} \cdot \exists t_1, \ldots, t_n \in S \cdot [\![\mathbb{R}(s,t_1) \wedge \varphi]\!](t_1) \sqsupseteq \ell_1 \wedge \cdots \wedge \\ [\![\mathbb{R}(s,t_n) \wedge \varphi]\!](t_n) \sqsupseteq \ell_n \wedge \bigsqcup_{1 \le i \le n} \ell_i \sqsupseteq \ell\end{array}}\ \text{one-point rule}}{[\![EX\varphi]\!](s) \sqsupseteq \ell}\ EX$$

(b)

$$\frac{[\![\mathbb{R}(s,t'_1) \wedge \varphi]\!](t'_1) \sqsupseteq \ell_1 \qquad \cdots \qquad [\![\mathbb{R}(s,t'_n) \wedge \varphi]\!](t'_n) \sqsupseteq \ell_n}{[\![EX\varphi]\!](s) \sqsupseteq \ell}$$

(c)

$$\begin{tikzcd}
[\![\mathbb{R}(s,t'_1) \wedge \varphi]\!](t'_1) \sqsupseteq \ell_1 & \cdots & [\![\mathbb{R}(s,t'_n) \wedge \varphi]\!](t'_n) \sqsupseteq \ell_n \\
& {[\![EX\varphi]\!](s) \sqsupseteq \ell} \arrow[ul, "\mathbb{R}(s,t'_1)"] \arrow[u] \arrow[ur, "\mathbb{R}(s,t'_n)"'] &
\end{tikzcd}$$

(d)

$$\begin{tikzcd}
t'_1 & \cdots & t'_n \\
& s \arrow[ul, "\mathbb{R}(s,t'_1)"] \arrow[u] \arrow[ur, "\mathbb{R}(s,t'_n)"'] &
\end{tikzcd}$$

Fig. 10. From proofs to witnesses.

4 Witness and Counterexample Generation

In this section, we describe how to use the proof system introduced in Section 3 to generate witnesses and counterexamples for multi-valued model-checking. We also discuss and illustrate the tool support for this approach.

4.1 From Proofs to Witnesses and Counterexamples

In this section we show how to extract a witness to $[\![\varphi]\!](s) = \ell$, where $\ell \in \mathcal{L}$, from a proof of $[\![\varphi]\!](s) \sqsupseteq \ell$. Counterexamples are extracted in a similar way.

The general form of the proof of validity of $[\![EX\varphi]\!](s) \sqsupseteq \ell$ is shown in Figure 10(a). This proof corresponds to a witness for EX, namely, a tree rooted at s, with children

$t_1, \ldots, t_n$, where an edge from s to t_i is labeled by the value of $\mathbb{R}(s, t_i)$. This correspondence between the proof an the witness suggests a simple procedure for extracting the witness from the proof:

1. remove all nodes from the proof tree except for (a) the root node, and (b) nodes that result from the application of the one-point rule (see Figure 10(b));
2. replace horizontal bars by directed edges, and label each edge incoming into a node $[\![\mathbb{R}(s, t_i) \wedge \varphi]\!](t_i)$ by the value of $\mathbb{R}(s, t_i)$ (see Figure 10(c));
3. relabel the top node by s, and each node of the form $[\![\mathbb{R}(s, t_i) \wedge \varphi]\!](t_i)$ by t_i (see Figure 10(d)).

The result is a tree rooted at s, with successor states $t'_1 ... t'_n$ — the witness for EX.

In general, the proof tree for a statement $[\![\varphi]\!](s) \sqsupseteq \ell$ can be partitioned into proof nodes that result from the application of the one-point rule to EX (and thus correspond directly to a step in the χKripke structure); and the "glue" that binds these steps into a complete witness. The witness can be extracted from the proof tree by the same procedure as in the EX case.

4.2 Tool Support

As discussed above, extracting witnesses from proofs effectively amounts to hiding certain proof steps, so proofs and witnesses are just two extremes in the presentation of the reasons behind the result of the model-checker to the user. The trade-off here is between size/complexity and ease of use. Proofs exhibit all of the reasoning steps explicitly, and make it easy for the user to follow each step. However, this excessive verbosity makes proofs much larger than corresponding witnesses and counterexamples. On the other hand, witnesses and counterexamples require the user to have a more detailed knowledge of the model-checking algorithm and the underlying De Morgan algebra.

To leverage the advantages of both presentations, we have developed an interactive witness browser tool — KEGVis [12]. Internally, the tool uses the proof view of the witness, while providing the user with the witness view.

A snapshot of KEGVis, showing the proof for $\varphi = ((EFEG\ r) \sqsupseteq \text{TT})$ on the χKripke structure in Figure 2 using the algebra $\mathbf{2x2}$, is given in Figure 11. Initially, the user is presented with the witness view of the proof, indicated by double-line nodes and arrows in the graph. Each state of the χKripke structure is represented by values of its atomic propositions, and for conciseness only the propositions that change from one state to the next are shown. For example, in the initial state s_0, $r = \text{FF}$. In its left successor, s_2, the value of r stays the same, so it is not shown in Figure 11. Additionally, each node is labeled with the χCTL($\mathcal{L}$) formula whose proof depends on that state. The "scissors" symbol next to a graph node indicates that a subgraph emanating from this node is hidden. Thus, the proof of $EFEG\ r$ is available in the initial state, and the proof of $E[\top\ U_1\ EG\ r]$ — in its left and right successors. The user of the tool can navigate through the witness or explore different proof parts by expanding them. For example, in the scenario given in Figure 11, the user has decided to fully expand the proof attached to the root node.

The information gathered from the proof can be used to gain better understanding and support navigation through the witness. For example, examining the proof attached

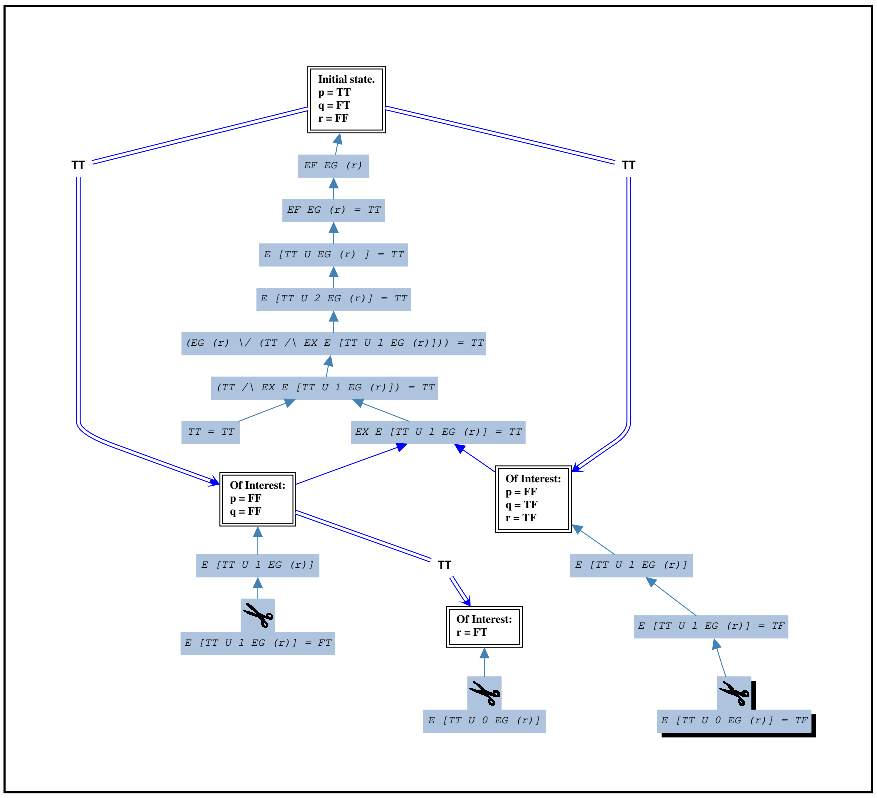

Fig. 11. Screenshot of KEGVis.

to the root node, we see that our original formula is simplified to $E[\top\ U_2\ EG\ r]$, which means that each path in the witness must be of length at most three. The reader is referred to [12] for a more comprehensive description of different browsing strategies.

4.3 Optimality, Minimality, and Finding the "Best" Witness

So far, we have concentrated on developing a sound and complete proof system for χCTL. We have also shown how to efficiently generate such proofs, and exhibited a one-to-one correspondence between proofs and witnesses. To simplify our reasoning, we have tried to use the simplest proof rules possible. The downside of this technique is that it treats all valid witnesses as equivalent.

In the rest of this section, we explore how a particular user objective, i.e. finding the smallest witness, can be met by introducing additional proof rules into our system. For brevity, we only concern ourselves with witnesses for χECTL.

We start by identifying the points in the proof where an improvement is possible. In general, such points correspond to branches in the proof tree, and in the case of witnesses for χECTL, they are identified by a disjunction: (a) a disjunction that occurs

$$\dfrac{\llbracket p\rrbracket(s_0) \sqsupseteq \mathrm{TT} \qquad \llbracket q\rrbracket(s_0) \sqsupseteq \mathrm{FT} \qquad \mathrm{TT} \sqcup \mathrm{FT} \sqsupseteq \mathrm{TT}}{\llbracket p \vee q\rrbracket(s_0) \sqsupseteq \mathrm{TT}}\ \vee\text{-rule}$$

Fig. 12. A proof of $\llbracket p \vee q\rrbracket(s_0)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\llbracket r\rrbracket(s_1) \sqsupseteq \mathrm{TF} \qquad \dfrac{\dfrac{\llbracket r\rrbracket(s_1) \sqsupseteq \mathrm{TF}}{\llbracket E[\top\ U_0\ r]\rrbracket(s_1) \sqsupseteq \mathrm{TF}}\ EU_0}{\llbracket EXE[\top\ U_0\ r]\rrbracket(s_1) \sqsupseteq \mathrm{TF}}\ EX}{\llbracket r \vee EXE[\top\ U_0\ r]\rrbracket(s_1) \sqsupseteq \mathrm{TF}}\ \vee\text{-rule}}{\llbracket E[\top\ U_1\ r]\rrbracket(s_1) \sqsupseteq \mathrm{TF}}\ EU_i \qquad \begin{array}{c}\llbracket r\rrbracket(s_3) \sqsupseteq \mathrm{FT}\\ \vdots \\ \llbracket E[\top\ U_1\ r]\rrbracket(s_2) \sqsupseteq \mathrm{FT}\end{array}}{\llbracket E[\top\ U_2\ r]\rrbracket(s_0) \sqsupseteq \mathrm{TT}}\ EX}{\llbracket E[\top\ U\ r]\rrbracket(s_0) \sqsupseteq \mathrm{TT}}\ EU$$

Fig. 13. Partial proof of $\llbracket E[\top\ U\ r]\rrbracket(s_0)$.

explicitly in a χECTL formula, and (b) a disjunction that is implicitly introduced by the EX operator.

Consider the proof of $P = \llbracket p \vee q\rrbracket(s_0) \sqsupseteq$ TT for the χKripke structure in Figure 2. The proof is shown in Figure 12. Clearly, since $\llbracket p\rrbracket(s_0) =$ TT, it is sufficient to justify P; however, our current $\vee$-rule requires us to prove the fact that $\llbracket q\rrbracket(s) \sqsupseteq$ FT as well. We solve this problem by strengthening the $\vee$-rule as follows:

$$\dfrac{\exists \ell_1 \cdot \llbracket \varphi\rrbracket(s) \sqsupseteq \ell_1 \ \wedge\ \ell_1 \sqsupseteq \ell}{\llbracket \varphi \vee \psi\rrbracket(s) \sqsupseteq \ell}\ \vee\text{-rule}$$

This ensures that the resulting proof does not contain any unnecessary branches.

Next, consider the proof for $\llbracket E[\top\ U\ r]\rrbracket(s_0) \sqsupseteq$ TT on the χKripke structure in Figure 2. A fragment of this proof is given in Figure 13. The problematic leaf is the one labeled with

$$\llbracket r \vee EXE[\top\ U_0\ r]\rrbracket(s_1) \sqsupseteq \mathrm{TF}$$

Since $\llbracket r\rrbracket(s_1) =$ TF and $\llbracket EXE[\top\ U_0\ r]\rrbracket(s) =$ TF, the proof generator non-deterministically chooses the subformula to expand. Thus, it can potentially decide to expand the EU formula, leading to a longer witness. This particular problem is solved by introducing an additional heuristic, telling the proof generator to always resolve non-deterministic choices by picking the smallest formula. However, in general, it is not possible to predict the size of the witness based solely on the syntax of the formula. A choice of a good heuristic typically depends on the additional domain and model knowledge. We leave the exploration and evaluation of various heuristics possible in this case for future work.

Finally, we consider the EX operator. In Section 3.3 we have shown that the breadth of the witness at the point where the one-point rule is applied to EX is determined by the size of the set $U' = \{\ell_i \mid \exists t \in S \cdot img(t) = \ell_i\}$. However, this solution is not optimal. For example, if $U' = \{\text{TT}, \text{TF}, \text{FT}\}$, then our witness for EX, given by the set U in Section 3.3, contains three states: t_1 with $img(t_1) = \text{TT}$, t_2 with $img(t_2) = \text{TF}$, and t_3 with $img(t_3) = \text{FT}$. However, it is sufficient to use either $\{t_1\}$ or $\{t_2, t_3\}$ since $img(t_1) = \text{TT}$ and $img(t_2) \sqcup img(t_3) = \text{TT}$. Thus, we can use one of the join-irredundant subsets of U' for the application of the one-point rule, instead of using U' directly.

5 Conclusion and Related Work

In this paper, we presented a technique for witness and counterexample generation for multi-valued model-checking. This technique is based on the concept of *proof-like* counterexamples, introduced in [12]. In fact, the automated proof-generation of Section 3.3 can be seen as simulating a run of a local tableau-based model-checker [18], where the information collected from a run of a global model-checker is used to guide the construction of the proof. However, unlike Stevens et al. [18], we restrict our attention to χCTL($\mathcal{L}$) for the given algebra $\mathcal{L}$, and use the insights provided by the counterexample generation algorithm of Clarke et al. [8, 9] to derive a specialized EG-rule.

The automated proof-generation algorithm presented in Section 3.3 makes use of a model-checker as a decision procedure. Alternatively, the same information can be extracted from the support sets of Tan and Cleaveland [19], or deductive proofs of Namjoshi [14]. This makes it possible to use the technique presented here for interactive unrolling of deductive proofs (and support sets) into witnesses and counterexamples.

In this paper, we have concentrated on the technical issues surrounding the counterexample and witness generation for multi-valued model-checking. We have only briefly discussed the potential of introducing an ordering on witnesses and identifying the best or the most interesting witness. The proof-like representation of witnesses and counterexamples allows us to define a number of strategies for their navigation and exploration. For example, they allow the user to specify starting and stopping conditions so that he/she can navigate to the "interesting" part of the witness. Other examples include preferring the state with maximum/minimum number of successors, choosing step granularity, forward/ backward exploration, etc. We have also built a tool KEGVis [11] for visualization and interactive exploration of witnesses and counterexamples. A partial list of strategies together with the description of KEGVis appear in [12]. A more comprehensive treatment is in [5].

References

1. N.D. Belnap. "A Useful Four-Valued Logic". In Dunn and Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 30–56. Reidel, 1977.
2. G. Bruns and P. Godefroid. "Temporal Logic Query-Checking". In *Proceedings of 16th Annual IEEE Symposium on Logic in Computer Science (LICS'01)*, pages 409–417, Boston, MA, USA, June 2001. IEEE Computer Society.
3. M. Chechik, B. Devereux, and A. Gurfinkel. "χChek: A Multi-Valued Model-Checker". In *Proceedings of 14th International Conference on Computer-Aided Verification (CAV'02)*, Lecture Notes in Computer Science, pages 505–509, Copenhagen, Denmark, July 2002. Springer.

4. M. Chechik, S. Easterbrook, and V. Petrovykh. "Model-Checking Over Multi-Valued Logics". In *Proceedings of Formal Methods Europe (FME'01)*, volume 2021 of *Lecture Notes in Computer Science*, pages 72–98. Springer, March 2001.
5. M. Chechik and A. Gurfinkel. "Exploring Counterexamples". In preparation, March 2003.
6. M. Chechik and A. Gurfinkel. "TLQSolver: A Temporal Logic Query Checker". Submitted of publication, January 2003.
7. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
8. E.M. Clarke, O. Grumberg, K.L. McMillan, and X. Zhao. "Efficient Generation of Counterexamples and Witnesses in Symbolic Model Checking". In *Proceedings of 32nd Design Automation Conference (DAC 95)*, pages 427–432, San Francisco, CA, USA, 1995.
9. E.M. Clarke, Y. Lu, S. Jha, and H. Veith. "Tree-Like Counterexamples in Model Checking". In *Proceedings of the Seventeenth Annual IEEE Symposium on Logic in Computer Science (LICS'02)*, pages 19–29, Copenhagen, Denmark, July 2002. IEEE Computer Society.
10. S. Easterbrook and M. Chechik. "A Framework for Multi-Valued Reasoning over Inconsistent Viewpoints". In *Proceedings of International Conference on Software Engineering (ICSE'01)*, pages 411–420, Toronto, Canada, May 2001. IEEE Computer Society Press.
11. A. Gurfinkel. "Multi-Valued Symbolic Model-Checking: Fairness, Counterexamples, Running Time". Master's thesis, University of Toronto, Department of Computer Science, October 2002.
12. A. Gurfinkel and M. Chechik. "Proof-like Counterexamples". In *Proceedings of 9th International Conference on Tools and Algorithms for the Cosntruction and Analysis of Systems (TACAS'03)*, April 2003. To appear.
13. S. C. Kleene. *Introduction to Metamathematics*. New York: Van Nostrand, 1952.
14. K. Namjoshi. "Certifying Model Checkers". In *Proceedings of 13th International Conference on Computer-Aided Verification (CAV'01)*, volume 2102 of *LNCS*. Springer-Verlag, 2001.
15. S. Owre, N. Shankar, and J. Rushby. "User Guide for the PVS Specification and Verification System (Draft)". Technical report, Computer Science Lab, SRI International, Menlo Park, CA, 1993.
16. H. Rasiowa. *An Algebraic Approach to Non-Classical Logics. Studies in Logic and the Foundations of Mathematics*. Amsterdam: North-Holland, 1978.
17. V. Sofronie-Stokkermans. "Automated Theorem Proving by Resolution for Finitely-Valued Logics Based on Distributive Lattices with Operators". *An International Journal of Multiple-Valued Logic*, 6(3-4):289–344, 2001.
18. P. Stevens and C. Stirling. "Practical Model-Checking using Games". In B. Steffen, editor, *Proceedings of the 4th International Conference on Tools and algorithms for the construction and analysis of systems (TACAS '98)*, volume 1384, pages 85–101, New York, NY, USA, 1998. Springer-Verlag.
19. L. Tan and R. Cleaveland. "Evidence-Based Model Checking". In *Proceedings of 14th Conference on Computer-Aided Verification (CAV'02)*, volume 2404 of *LNCS*, pages 455–470, Copenhagen, Denmark, July 2002. Springer-Verlag.