Extending Extended Vacuity

Arie Gurfinkel and Marsha Chechik

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

Abstract. There has been a growing interest in detecting whether a logic specification holds in the system *vacuously*. For example, a specification "every request is eventually followed by an acknowledgment" holds vacuously on those systems that never generate requests. In a recent paper, Armoni et al. have argued against previous definitions of vacuity, defined as sensitivity with respect to syntactic perturbation. They suggested that vacuity should be *robust*, i.e., insensitive to trivial changes in the logic and in the model, and is better described as sensitivity with respect to semantic perturbation, represented by universal propositional quantification. In this paper, we extend the above results by giving a formal definition of robust vacuity that allows us to define and detect vacuous satisfaction and vacuous failure for arbitrary CTL* properties, even with respect to multiple occurrences of subformulas. We discuss complexity of our approaches and study the relationship between vacuity and abstraction.

1 Introduction

Model-checking gained wide popularity as an automated technique for effective analysis of software and hardware systems. Yet a major problem in practical applications of model-checking is that a successful run of the model-checker does not necessarily guarantee that the intended requirement is satisfied by the system [3]. For example, the property "every request must be followed by an acknowledgment" holds vacuously in a system that is known to never produce a request. Industrial researchers at the IBM Haifa Research Laboratory observed that vacuity is a serious problem [3] and that "... typically 20% of specifications pass vacuously during the first formal verification runs of a new hardware design, and that vacuous passes always point to a real problem in either the design, or its specification, or the environment" [3]. Further justification is given by case studies conducted by Purandare and Somenzi [21].

A recent paper by Armoni et al. [1] provides an excellent summary of the work on vacuity detection. We summarize only a few aspects of this work here. Vacuity detection research progressed along two separate axes. One [4, 3, 18, 15] aims at increasing the scope of applicability of vacuity detection algorithms. This research deals with deciding vacuity for temporal logics, from subsets of CTL and LTL to CTL^*, looking at formulas with one occurrence or several occurrences [1] of a subformula, looking at vacuous satisfaction and vacuous failure of formulas, generating witnesses for non-vacuity, etc. Most of this work assumes a *syntactic* definition of vacuity, provided by Beer et al. [4]: φ is vacuous in a subformula ψ if ψ can be replaced by any temporal logic formula.

An orthogonal line of research, initiated by Armoni et al. [1], is to examine the *meaning* of vacuity. The authors specified vacuity via universal quantification, formally defining it as satisfying the expression $\forall x \cdot \varphi[\psi \leftarrow x]$. With this definition, several different interpretations of the domain of x yield different notions of vacuity. When x ranges over temporal logic formulas, this corresponds to syntactic vacuity of [3], called *formula* vacuity in [1]. When checking vacuity of LTL properties, as is the goal of [1], x can also range over the set of computations, resulting in *trace* vacuity. Alternatively, x can range over boolean functions over the statespace, resulting in *structure* vacuity.

Armed with these definitions, the authors faced the task of choosing the most desirable one, both in terms of computational tractability, but more importantly, in terms of its correspondence to the intuitive notion of vacuity. Such a notion should be *robust*, i.e., insensitive to changes in the design that do not relate to the formula. For example, a property "if p is true now, it will remain true in the next state" should not be affected by adding a proposition q to the model. Robustness also includes insensitivity to the specification language: a formula passing vacuously, e.g., in a language with only negation and conjunction, should not pass nonvacuously once the specification language is extended, e.g., by adding implication. Armoni at al. exemplify that neither structure nor formula vacuity are robust. Structure vacuity depends too much on the model, and is sensitive to trivial modifications that do not correspond to changes in the design. Formula vacuity depends on the syntax of the underlying logic. For example, [1] presents an LTL formula that can be considered both vacuous in the model, and non-vacuous when past operators are allowed. The authors argue that trace vacuity is robust, but since robustness is specified only informally, it is not clear whether that argument is correct. Armoni et al. also prove that the decision procedure for checking vacuous *satisfaction* for LTL formulas over trace vacuity is tractable, making it the most desirable definition.

What is the "right" definition of vacuity? Does it change as we transition from LTL properties to CTL^* and from vacuous satisfaction (i.e., checking formulas that are known to hold in the system) to vacuous failure? Is vacuity detection under this robust definition tractable? We address these questions in this paper. In particular, we formalize the notion of robust vacuity and use the quantified temporal logic formulation of Armoni et. al to extend semantic vacuity to branching temporal logics such as CTL^*. Like [1], our definition is applicable to subformulas of both pure and mixed polarity. Note that our extension is not trivial when we move from linear-time to branching-time logic, i.e., from traces to trees. Although quantification over trees has been studied by Kupferman [17], neither of the interpretations suggested in her paper are robust enough, and a different interpretation is needed.

We further show that in general, vacuity detection for CTL^* is expensive, but identify several important fragments of CTL^* for which vacuity detection, or at least detecting vacuous satisfaction, is no harder than model-checking. One of such fragments is checking vacuous satisfaction of ACTL^*, which subsumes results of [1].

Verification of even medium-size models is impossible without abstraction. Since vacuity checking should supplement every verification activity, what happens with vacuity under abstraction? In this paper, we prove that vacuity is preserved by abstraction. Further, we show a surprising fact: vacuity detection algorithm is *more precise* than traditional abstract model-checking, that is, it is sometimes possible to determine that a formula is vacuously satisfied by an abstract model, even if the result of abstract model-checking is inconclusive!

The rest of paper is organized as follows: after providing the necessary background in Section 2, we define robust vacuity in Section 3 and study the complexity of vacuity detection. In Section 4, we discuss how vacuity detection can be reduced to 3-valued model-checking and how various semantics for 3-valued model-checking [5, 6] correspond to different notions of vacuity. Section 5 studies the relationship between vacuity and abstraction. Section 6 concludes the paper.

2 Background

In this section, we fix the notation and give the necessary background on model-checking and quantified temporal logics.

Models. We denote the set of boolean values $\{\mathsf{true}, \mathsf{false}\}$ by $\mathbf{2}$. A model is a Kripke structure $K = (S, R, S_0, AP, I)$, where S is a finite set of states, $R : S \times S \to \mathbf{2}$ is a total transition relation, $S_0 \subseteq S$ is a set of initial states, AP is a set of atomic propositions, and $I : S \times AP \to \mathbf{2}$ is a labeling function, assigning a value to each atomic proposition $a \in AP$ in each state. A path π of K is an infinite sequence of states in which every consecutive pair of states is related by the transition relation. We denote a path starting at state s by π_s, and the set of all such paths by Π_s.

Temporal Logic. Computation Tree Logic CTL^* is a branching-time temporal logic constructed from propositional connectives, temporal operators X (next), U (until), F (future), and G (globally), and path quantifiers A (forall) and E (exists). CTL^* denotes the set of all state formulas $\varphi = c \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid A\psi \mid E\psi$, where c is an atomic proposition, and $\psi = \varphi \mid X\psi \mid \psi\ U\ \psi \mid F\psi \mid G\psi$ is a path formula. The formal semantics of CTL* is available in [8]. We write $||\psi||^K(\pi) = \mathsf{true}$, or $||\psi||^K(\pi)$, to mean that ψ is satisfied by a path π of K. For state formulas, $||\varphi||^K(s)$ denotes the value of φ in state s of K, and is defined as follows:

$$||A\psi||^K(s) \triangleq \bigwedge_{\pi_s \in \Pi_s} ||\psi||^K(\pi_s) \qquad\qquad ||E\psi||^K(s) \triangleq \bigvee_{\pi_s \in \Pi_s} ||\psi||^K(\pi_s)$$

φ holds in K, denoted $||\varphi||^K$, iff it holds in every initial state: $\bigwedge_{s_0 \in S_0} ||\varphi||^K(s_0)$.

We write $\varphi[x]$ to indicate that the formula φ may contain an occurrence of x, in which case x is *positive* (or of *positive polarity*) if it occurs under the scope of an even number of negations, and *negative* otherwise. A subformula x is *pure* in φ if all of its occurrences have the same polarity. We write $\varphi[x \leftarrow y]$ for a formula obtained from φ by replacing x by y. A formula φ is *universal* (or in ACTL^*) if all of its temporal path quantifiers are universal, and is *existential* (or in ECTL^*) if they are existential. In both cases, negation is only allowed at the level of atomic propositions.

Fragments of CTL^* in which all formulas are of the form $A\psi$, where ψ is a path formula, and those in which every occurrence of a path quantifier is immediately followed by a temporal operator, are called Linear Temporal Logic (LTL) and Computation Tree Logic (CTL), respectively.

Quantified Temporal Logic. Quantified Temporal Logic QCTL^* is an extension of CTL^* with universal and existential quantification over atomic propositions [17]. In this paper, we consider a fragment $\{\varphi, \forall x \cdot \varphi, \exists x \cdot \varphi \mid \varphi \in \mathrm{CTL}^*\}$, still referring to it as QCTL^*.

The syntax of QCTL^* does not restrict the domain of quantifiers. Thus, there are several different definitions of QCTL^* semantics with respect to a Kripke structure; we consider three of these: *structure* [17], *tree* [17], and *amorphous* [9].

Structure Semantics. Under this semantics, each free variable x is interpreted as a boolean function over the statespace, i.e., $x \in [S \to \mathbf{2}]$. For example, $\forall x \cdot \varphi$ is true in K under structure semantics if replacing x by an arbitrary boolean function results in a formula that is true in K. Formally, the semantics of a quantifier $Q \in \{\forall, \exists\}$ is:

$$||Qx \cdot \varphi||_s^K \triangleq Qx \in [S \to \mathbf{2}] \cdot ||\varphi[x]||^K \quad \text{(structure semantics)}$$

Alternatively, structure semantics can be understood as follows. For an atomic proposition x in K, let K_{-x} denote the result of removing x from K, i.e., $AP_{-x} = AP \setminus \{x\}$. We say that K' is an $\{x\}$*-variant* of K if K'_{-x} and K are isomorphic. A formula $\forall x \cdot \varphi[x]$ is satisfied by K under structure semantics iff $\varphi[x]$ holds in all $\{x\}$-variants of K.

Tree Semantics. Tree semantics of QCTL^* is defined on the computation tree $T(K)$ obtained by unrolling K from its initial state. Formally,

$$||Qx \cdot \varphi||_T^K \triangleq Qx \cdot ||\varphi[x]||_s^{T(K)} \quad \text{(tree semantics)}$$

That is, a formula $\forall x \cdot \varphi[x]$ is satisfied by K under tree semantics iff it is satisfied by every $\{x\}$-variant of the computation tree of K.

Amorphous Semantics. We start by revisiting the notions of simulation and bisimulation.

Definition 1. *[20] Let K and K' be Kripke structures with identical sets of atomic propositions AP. A relation $\rho \subseteq S \times S'$ is a simulation relation iff $\rho(s, s')$ implies that*

1. $\forall p \in AP \cdot I'(s', p) \Leftrightarrow I(s, p)$, *and*
2. $\forall t' \in S' \cdot R'(s', t') \Rightarrow \exists t \in S \cdot R(s, t) \wedge \rho(t, t')$

A state s *simulates* a state s' if $(s, s') \in \rho$. A Kripke structure K simulates K' iff every initial state of K' is simulated by an initial state of K. Simulation between K and K' preserves ACTL^*: for any $\varphi \in \mathrm{ACTL}^*$, $||\varphi||^K \Rightarrow ||\varphi||^{K'}$. K and K' are *bisimilar* iff there exists a simulation relation ρ between K and K' such that ρ^{-1} is a simulation between K' and K. The set of all structures bisimilar to K is denoted by $\mathcal{B}(K)$. Bisimulation preserves CTL^*: $\forall \varphi \in \mathrm{CTL}^* \cdot \forall K' \in \mathcal{B}(K) \cdot ||\varphi||^K \Leftrightarrow ||\varphi||^{K'}$.

Let K and K' be Kripke structures such that the set of atomic propositions of K' is $AP \cup \{x\}$. Then, K and K' are $\{x\}$*-bisimilar* iff K and K'_{-x} are bisimilar. The set of all $\{x\}$-bisimilar structures to K is denoted by $\mathcal{B}_x(K)$.

Amorphous semantics of QCTL^* is defined as follows:

$$||Qx \cdot \varphi[x]||_a^K \triangleq \forall K' \in \mathcal{B}_x(K) \cdot ||\varphi[x]||^{K'} \quad \text{(amorphous semantics)}$$

That is, a formula $\forall x \cdot \varphi[x]$ is satisfied by K under amorphous semantics iff $\varphi[x]$ is satisfied by every $\{x\}$-bisimulation of K.

For universally quantified formulas, amorphous semantics imply tree semantics, and tree semantics imply structure semantics; further, the implication is strict [9].

3 Extended Vacuity

In this section, we construct a "robust" definition of vacuity for temporal logic and study complexity of vacuity detection.

3.1 Defining Vacuity

As argued by Armoni et al. [1], vacuity is robust if (a) it is independent of the details of the logic used to formalize the property, e.g., if φ is deemed to be vacuous as a propositional formula, it must remain so in logics that subsume propositional logic, such as CTL; and (b) it is independent of a particular modeling of the system, i.e., the vacuity cannot be "fixed" by changing the model to an equivalent one.

We start our exploration of vacuity with propositional logic. A model of a propositional formula φ is simply a valuation of all atomic propositions of φ, and the value of each propositional formula is either true or false. Thus, we can check the dependence of φ on a subformula ψ by checking whether replacing ψ by true and false affects the value of φ.

Definition 2. *A propositional formula φ is* vacuous *in a subformula ψ, or simply ψ-vacuous, in a model K iff replacing ψ by* true *and* false *does not affect the value of φ:* $||\varphi[\psi \leftarrow \mathsf{true}]||^K = ||\varphi[\psi \leftarrow \mathsf{false}]||^K$.

Vacuity of a propositional formula in a model K can be expressed as validity of a quantified boolean formula in K; that is, φ is ψ-vacuous iff $\forall x \in \mathbf{2} \cdot ||\varphi[\psi \leftarrow x]||^K$ or $\forall x \in \mathbf{2} \cdot ||\neg\varphi[\psi \leftarrow x]||^K$. Note that this definition ensures that vacuity of a formula φ in a model K does not change if we extend K with new atomic propositions, or allow for new logical connectives, such as implication.

It may seem that Definition 2 describes robust vacuity for temporal logic as well, but it is not the case. For example, consider the formula $\varphi = (AXp) \vee (AX\neg p)$ that formalizes the property "p changes deterministically". According to our intuition, φ expresses a "reasonable" requirement and is not vacuous in any model. Yet in any model both $(AX\mathsf{true}) \vee (AX\neg\mathsf{true})$ and $(AX\mathsf{false}) \vee (AX\neg\mathsf{false})$ evaluate to true, which makes φ vacuous according to Definition 2.

The problem is that the interpretation of a temporal formula in a model is its value in *every* state of the model. Therefore, replacing a subformula by true and false is not sufficient for identifying whether the subformula is important. Following this, we extend the definition of vacuity to account for all boolean functions over the statespace of the model, i.e., $S \to \mathbf{2}$. The resulting definition was introduced in [1] as *structure* vacuity.

Definition 3. *[1] A temporal logic formula φ is* structure *ψ-vacuous in a model K iff* $\forall x \in [S \to \mathbf{2}] \cdot ||\varphi[\psi \leftarrow x]||^K$ *or* $\forall x \in [S \to \mathbf{2}] \cdot ||\neg\varphi[\psi \leftarrow x]||^K$.

Once again, Definition 3 is not strong enough to capture our intuition, because it makes vacuity dependent on a particular model of the system. For example, consider the two models in Figure 1, both describing a system in which p holds along every execution. According to Definition 3, the formula $\varphi = (AXp) \vee (AX\neg p)$ is p-vacuous in the model in Figure 1(a), but is not p-vacuous in the model in Figure 1(b).

As indicated by the example, it is not sufficient to define vacuity with respect to a *particular* model K. Instead, it should also consider any model that is equivalent to K.

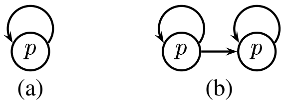

Fig. 1. Two models of a system in which p holds along every execution.

For temporal logic, two models are considered to be equivalent iff they are bisimilar, which leads us to the final definition of vacuity.

Definition 4. *A temporal logic formula φ is ψ-vacuous in a Kripke structure K iff it is structure ψ-vacuous in K and is ψ-vacuous in any Kripke structure K' that is bisimilar to K. That is, $\forall K' \in \mathcal{B}(K) \cdot \forall x \in [S' \to \mathbf{2}] \cdot ||\varphi[\psi \leftarrow x]||^{K'}$, or $\forall K' \in \mathcal{B}(K) \cdot \forall x \in [S' \to \mathbf{2}] \cdot ||\neg\varphi[\psi \leftarrow x]||^{K'}$.*

Here, S' is the statespace of K'. Definition 4 closely matches the amorphous semantics for quantified temporal logic [9]. Thus, checking vacuity can be reduced to checking validity of a quantified temporal logic formula: φ is ψ-vacuous in a Kripke structure K iff $||\forall x \cdot \varphi[\psi \leftarrow x]||_a^K$ or $||\forall x \cdot \neg\varphi[\psi \leftarrow x]||_a^K$. That is, $\varphi[\psi \leftarrow x]$ is either satisfied or violated in every model that is $\{x\}$-bisimilar to K. This definition is independent of any particular logic. For example, a propositional formula is propositionally vacuous iff it is vacuous by Definition 4; a CTL formula is vacuous iff it is vacuous as a CTL^* formula, etc. Note that Definition 4 is bisimulation-closed, and is therefore insensitive to different encodings of the model.

We conclude this section by summarizing some of the key properties of vacuity. The proofs are available in the extended version of the paper.

Theorem 1. *Let φ be a temporal logic formula, ψ be any non-constant proper subformula of φ, and K be a Kripke structure. Then,*

1. *if φ is valid or unsatisfiable, then φ is vacuous in any model;*
2. *if ψ is valid or unsatisfiable, then φ is ψ-vacuous in any model;*
3. *if φ is ψ-vacuous in K, and K' is bisimilar to K, then φ is ψ-vacuous in K';*
4. *if φ is ψ-vacuous in K, and K' is an arbitrary Kripke structure, then φ is ψ-vacuous in $K||K'$ (synchronous parallel composition of K and K').*

3.2 Complexity of Vacuity Detection

In this section, we study the complexity of detecting vacuity for temporal logics CTL and CTL^*.

In general, detecting vacuity for CTL and CTL^* is in the same complexity class as the satisfiability problem for these logics.

Theorem 2. *The complexity of detecting whether a formula φ is ψ-vacuous is EXPTIME-complete for CTL, and 2EXPTIME-complete for CTL^*.*

The proof of this and most other theorems in this paper is available in the Appendix.

Although vacuity detection for an arbitrary CTL^* formula is computationally expensive, we identify several important fragments for which vacuity detection is in the same complexity class as model-checking.

Syntactically Monotone Formulas. A formula φ is *monotonically increasing* in a subformula ψ iff $(x \Rightarrow y) \Rightarrow (\varphi[\psi \leftarrow x] \Rightarrow \varphi[\psi \leftarrow y])$, and is *monotonically decreasing* if $(x \Rightarrow y) \Rightarrow (\varphi[\psi \leftarrow x] \Leftarrow \varphi[\psi \leftarrow y])$; furthermore, φ is said to be *monotone* in ψ iff it is either monotonically increasing or decreasing. A formula φ is *syntactically monotone* in ψ iff ψ occurs with pure polarity in φ. The complexity of detecting whether a formula φ, monotone in ψ, is also ψ-vacuous is the same as the complexity of model-checking φ:

Theorem 3. *If a temporal logic formula φ is monotone in a subformula ψ, then the complexity of detecting whether φ is ψ-vacuous in a Kripke structure K is $2 \times MC(\varphi)$, where $MC(\varphi)$ is the complexity of model-checking φ in K.*

Unfortunately, deciding whether a formula is monotone in a subformula is as hard as detecting vacuity.

Theorem 4. *The complexity of deciding whether a formula φ is monotone in a subformula ψ is EXPTIME-hard for CTL, and 2EXPTIME-hard for* CTL^*.

Although deciding whether a formula is monotone is expensive, syntactic monotonicity is easy to establish, and it guarantees monotonicity.

Theorem 5. *If φ is a CTL^* formula that is syntactically monotone in ψ, then φ is also monotone in ψ.*

Thus, deciding vacuity for a CTL^* formula in a subformula of pure polarity, and therefore syntactically monotone, is of the same complexity as model-checking.

True ACTL^* Formulas and False ECTL^* Formulas. Here, we show that for ACTL^* formulas that are known to be true in a given model, vacuity detection is as cheap as model-checking. Dually, the complexity of detecting vacuity of ECTL^* formulas that are known to be violated by the model is also as cheap as model-checking.

For a given Kripke structure K, we define $K_x = (AP_x, S \times \{0,1\}, S_0 \times \{0,1\}, R_x, I_x)$, where $AP_x = AP \cup \{x\}$; $I_x(\langle s,i\rangle, p) = I(s,p)$; $I_x(\langle s,i\rangle, x) = \mathsf{true} \Leftrightarrow i = 1$; and $R_x(\langle s,i\rangle, \langle t,j\rangle) \Leftrightarrow R(s,t)$. K_x is a synchronous parallel composition of K with a Kripke structure containing a single non-deterministic atomic proposition x. Note that K_x is $\{x\}$-bisimilar to K, where the bisimulation is given by $\rho_x = \{(s, \langle s,i\rangle) \mid i \in \{0,1\}\}$. Moreover, any K' that is $\{x\}$-bisimilar to K is simulated by K_x.

Theorem 6. *Let K be a Kripke structure, and K_x be as described above. Then, if K' is $\{x\}$-bisimilar to K, then it is simulated by K_x.*

Since simulation preserves ACTL^*, vacuity detection for an arbitrary ACTL^* formula is reducible to model-checking over K_x. Note that this also proves that our definition of vacuity for LTL is equivalent to trace vacuity of Armoni et al [1].

Theorem 7. *Let φ be an ACTL^* formula, ψ be a subformula of φ, and K be a Kripke structure. If φ is known to be satisfied by K, then complexity of detecting vacuity of φ in ψ is $2 \times MC(\varphi)$.*

Note that the statespace of K_x is double of K. However, K_x does not impose any restrictions on the atomic proposition x; therefore, the symbolic representation of the transition relation of K_x is identical to that of K!

Reducing CTL^* Formulas. For a fixed model K, any temporal formula φ is logically equivalent to a propositional formula over the atomic propositions of K, denoted $Prop(\varphi, K)$. If ψ is a state subformula of φ, then replacing ψ by its propositional equivalent $Prop(\psi, K)$ does not affect the value of φ in K, and any model bisimilar to K. That is,

$$\forall K' \in \mathcal{B}(K) \cdot ||\varphi||^{K'} = ||\varphi[\psi \leftarrow Prop(\varphi, K')]||^{K'} \quad \text{(propositional substitution)}$$

For example, if ψ is true in a state s iff the propositional formula $p \wedge q$ is true in s, then $Prop(\psi, K) = p \wedge q$, and every occurrence of ψ can be replaced by $p \wedge q$.

Propositional substitution preserves vacuity, in the sense that a formula φ is ψ-vacuous iff a formula φ', obtained by replacing all subformulas that do not contain ψ with their propositional equivalents, is ψ-vacuous.

Theorem 8. *Let $\varphi[x, \psi]$ be an CTL^* formula, with a subformula x and a* state *subformula ψ, and K be a Kripke structure. Then, φ is x-vacuous iff $\varphi[x, \psi \leftarrow Prop(\psi, K)]$ is x-vacuous.*

This theorem can be used to reduce some CTL^* formulas to ACTL^*, thus reducing the complexity of vacuity detection for them. In particular, if a subformula ψ of φ occurs in the scope of only universal path quantifiers, replacing all other state subformulas of φ reduces φ to an ACTL^* formula.

Theorem 9. *Let φ be a CTL^* formula, ψ be a subformula of φ that occurs in the scope of only universal path quantifiers. Then, if a Kripke structure K satisfies φ, the complexity of detecting vacuity of φ in ψ is $2 \times MC(\varphi)$.*

4 Vacuity Detection via 3-valued Model Checking

In this section, we show that 3-valued model-checking provides a uniform framework for studying vacuity detection.

4.1 3-Valued Model-Checking

A 3-valued Kleene logic **3** [16] is an extension of a classical two-valued logic of true and false, with an additional value maybe, representing uncertainty. Propositional operators in the logic are defined via the *truth* ordering $\sqsubseteq$, where true $\sqsubseteq$ maybe $\sqsubseteq$ false. Intuitively, $a \sqsubseteq b$ indicates that a is *less true* than b. Conjunction and disjunction are given by meet (minimum) and join (maximum) operators of the truth ordering, respectively. Negation is defined as: $\neg$true $=$ false, $\neg$false $=$ true, and $\neg$maybe $=$ maybe. Kleene logic preserves most of the laws of classical logic, such as De Morgan laws ($\neg(a \wedge b) = \neg a \vee \neg b$), and an involution of negation ($\neg\neg a = a$), but not the laws of excluded middle ($a \vee \neg a =$ true) and non-contradiction ($\neg a \wedge a =$ false). The values of Kleene logic can also be ordered according to the *information* ordering $\preceq$, where maybe $\preceq$ true and maybe $\preceq$ false. That is, maybe contains the least amount of information, whereas true and false are incomparable.

Model-checking is extended to the 3-valued logic by allowing models with uncertain information, represented by atomic propositions with value maybe. Formally, a model is a 3-valued Kripke structure $K = (AP, S, S_0, R, I)$, where AP, S, S_0, and R are classical, and I is a function $S \times A \to \mathbf{3}$. Any classical Kripke structure is simply a 3-valued Kripke structure that does not make use of maybe values. Adding 3-valued transitions to 3-valued Kripke structures does not increase their expressive power [12].

Semantics of CTL^* is extended to 3-valued models by interpreting propositional operators as Kleene, and referred to as *compositional*. For example, $||EXp||(s) = \bigvee_{t \in S} R(s,t) \wedge ||p||(t)$. In classical models, the 3-valued and two-valued semantics of CTL^* coincide.

The information ordering $\preceq$ is extended to 3-valued Kripke structures providing a completeness preorder that connects classical and 3-valued Kripke structures. Intuitively, a 3-valued Kripke structure K is more complete than K', written $K' \preceq K$, if the information contained in K' is less certain than that in K. In this case, K is a *refinement* of K'.

Definition 5. *[5] A relation $\rho \subseteq S \times S'$ is a refinement between 3-valued Kripke structures K and K' iff $\rho(s,s')$ implies*

1. $\forall p \in AP \cdot I(s,p) \preceq I'(s',p)$
2. $\forall t \in S \cdot R(s,t) \Rightarrow \exists t' \in S' \cdot R'(s',t') \wedge \rho(t,t')$
3. $\forall t' \in S' \cdot R'(s',t') \Rightarrow \exists t \in S \cdot R(s,t) \wedge \rho(t,t')$

A state s is refined by a state s' ($s \preceq s'$) if there exists a refinement ρ containing (s,s'). A Kripke structures K is refined by K' ($K \preceq K'$) if there exists a refinement ρ relating their initial states: $\forall s \in S_0 \cdot \exists s' \in S'_0 \cdot \rho(s,s')$ and $\forall s' \in S'_0 \cdot \exists s \in S_0 \cdot \rho(s,s')$. Intuitively, K refines K' if K is bisimilar to K', provided we ignore the uncertain (maybe) atomic propositions of K'. In particular, bisimulation and refinement coincide on classical structures.

Theorem 10. *[5] For 3-valued Kripke structures K and K' and a CTL^* formula φ, $K \preceq K'$ implies $||\varphi||^K \preceq ||\varphi||^{K'}$, i.e., refinement preserves 3-valued CTL^*.*

The refinement relation relates 3-valued and classical models. For a 3-valued Kripke structure K, let $\mathcal{C}(K)$ denote the set of *completions* of K – the set of all classical Kripke structures that refine K. For any completion $K' \in \mathcal{C}(K)$, the structure K can be seen as less precise than K' in the sense that any CTL^* formula that evaluates to a definite value (either true or false) in K, evaluates to the same value in K'. That is, for any $K' \in \mathcal{C}(K)$ and any CTL^* formula φ, $(||\varphi||^K = \mathsf{true}) \Rightarrow (||\varphi||^{K'} = \mathsf{true})$ and $(||\varphi||^K = \mathsf{false}) \Rightarrow (||\varphi||^{K'} = \mathsf{false})$. The converse, however, does not hold is in general. For example, the formula $p \vee \neg p$ is true in any classical model, yet its value is maybe in any state of a 3-valued model that assigns maybe to p (i.e., $p \vee \neg p = \mathsf{maybe} \vee \neg\mathsf{maybe} = \mathsf{maybe} \vee \mathsf{maybe} = \mathsf{maybe}$).

To address this imprecision of compositional semantics, Bruns and Godefroid [6] have introduced an alternative semantics, calling it *thorough*.

Definition 6. *[6] Let K be a 3-valued Kripke structure, and φ a CTL^* formula. The value of φ in K under thorough semantics $||\varphi||^K_t$ is:* true *iff φ holds in every completion of K,* false *if it is* false *in every completion, and* maybe *otherwise.*

Thorough semantics always produces more exact answers than compositional, i.e., for any 3-valued Kripke structure K and a CTL^* formula φ, $||\varphi||^K \preceq ||\varphi||^K_t$. This additional precision comes at a price of complexity: model-checking a given branching temporal logic formula under compositional semantics is in the same complexity class as classical model-checking; however, its model-checking complexity under thorough semantics is as hard as satisfiability [6].

4.2 Vacuity Detection via 3-valued Model-Checking

In this section, we show that vacuity detection is closely related to 3-valued model-checking under thorough semantics.

Let K be a Kripke structure, and let K_x be a 3-valued Kripke structure obtained from K by: (a) adding to K a new atomic proposition x, and (b) setting the value of x to maybe in every state. The set $\mathcal{C}(K_x)$ of completions of K_x is equivalent to the set of Kripke structures which are $\{x\}$-bisimilar to K.

Theorem 11. *Let K be a Kripke structure, and K_x be as described above. Then, $K' \in \mathcal{C}(K_x)$ iff K' is $\{x\}$-bisimilar to K.*

This theorem suggests the following reduction from vacuity detection to 3-valued model-checking:

To check whether a formula φ is ψ-vacuous in K:
  Construct a 3-valued Kripke structure K_x.
  Model-check $\varphi[\psi \leftarrow x]$ in K_x under thorough semantics.
  if $||\varphi[\psi \leftarrow x]||_t^{K_x} \in \{\mathsf{true}, \mathsf{false}\}$
    then return "φ is ψ-vacuous"
  else return "φ is not ψ-vacuous".

The reduction in the other direction, i.e., from 3-valued model-checking to vacuity detection, is also possible, but is outside the scope of this paper.

As the result of this reduction, any advance in 3-valued model-checking is also applicable to vacuity detection. For example, the complexity of thorough semantics model-checking for CTL and CTL^* is known to be EXPTIME- and 2EXPTIME-complete [6], respectively, which provides an alternative proof of the first part of Theorem 2. For another notable example, consider ACTL^* persistence properties [19], i.e., properties of the form $AFGp$. These correspond to co-Büchi automata, and the complexity of model-checking these properties under thorough semantics is linear in the size of the model [11]. Thus, for such properties, the complexity of vacuity detection is the same as that of model-checking.

Note that since thorough semantics is more precise than compositional, model-checking $\varphi[\psi \leftarrow x]$ in K_x under compositional semantics yields a sound approximation to vacuity detection. If $||\varphi[\psi \leftarrow x]||^{K_x}$ is either true or false, then φ is ψ-vacuous; otherwise, the result of the model-checking is maybe, which gives no information about vacuity. Moreover, our previous work [14] showed that if ψ occurs in φ with pure polarity, then compositional semantics reduces to checking whether $||\varphi[\psi \leftarrow \mathsf{true}]||^K = ||\varphi[\psi \leftarrow \mathsf{false}]||^K$. Thus, by Theorems 3 and 5, compositional semantics provides a sound and complete algorithm for vacuity detection in subformulas of pure polarity.

An additional benefit of the reduction enables the use of 3-valued model-checkers, which are arguably necessary for combining model-checking and abstraction [11, 22], to solve the vacuity detection problem. Moreover, witnesses and counterexamples, produced by such model-checkers, can be used to explain why a given formula is deemed to be (non-)vacuous. To our knowledge, a 3-valued model-checker with thorough semantics has not been implemented. Further, we are not aware of an algorithm for generating witnesses and counterexamples in this case, but it appears to be equivalent to logic synthesis and thus very expensive. However, compositional 3-valued model-checkers, such as χChek [7], have been implemented, and their witnesses and counterexamples have a natural correspondence to witnesses for non-vacuity [4]. This connection has been explored in our previous work [15].

Finally, reducing vacuity detection to 3-valued model-checking makes it easier to explore the combination of vacuity detection with various abstraction techniques, as we explore in the next section.

5 Vacuity and Abstraction

One of the biggest obstacles in the use of model-checking is the statespace explosion problem – the size of the model doubles with addition of each new atomic proposition. Abstraction appears to be the most effective technique to combat this problem. In this section, we show how vacuity detection can be combined with abstraction techniques for model-checking.

5.1 Abstraction and 3-Valued Model-Checking

The key idea of abstraction is that instead of model-checking a property φ on a (concrete) model K_c, one first constructs an abstraction K_α of K_c, for example, by removing some atomic propositions of K_c, and then checks φ in K_α.

An abstraction is *sound* with respect to a set of properties Φ, if a definite value for any property $\varphi \in \Phi$ in an abstract model K_α, means that φ has the same value in the corresponding concrete model K_c. For example, if φ is true in K_α, it is true in K_c. An abstraction is called *exact* if it is sound with respect to CTL^*, and the result of model-checking any φ on the abstract model is always definite. For example, cone of influence and symmetry reductions are exact [8]. Note that if K_α is an exact abstraction of K_c, then K_α and K_c are bisimilar.

3-valued models provide a natural representation for abstract models. A 3-valued (or possibly classical) Kripke structure K_α is an abstraction of K_c iff K_c refines K_α, i.e., $K_\alpha \preceq K_c$. Model-checking a property φ in K_c can be done by model-checking φ in K_α under compositional semantics. If the value of φ in K_α is maybe, then nothing is known about its value in the concrete model. However, by Theorem 10, if φ evaluates to either true or false in K_α, then it has the same value in K_c. This guarantees that the abstraction is sound with respect to CTL^*. Common abstraction techniques, such as predicate [13] and Cartesian [2] abstractions, can be cast as 3-valued model-checking problems [10].

In the next section, we extend our definition of vacuity, as well as algorithms for its detection, to 3-valued models, which illustrates how vacuity detection can be combined with most common abstraction techniques.

5.2 Combining Vacuity Detection and Abstraction

Let K_α be a 3-valued Kripke structure. A definition of vacuity in an abstract model should be sound with respect to the abstraction used. That is, if a formula φ is vacuous in K_α, then it must be vacuous in all concretizations of K_α. Note that the set of concretizations of K_α is the same as the set $\mathcal{C}(K_\alpha)$ of its completions. This leads us to the following definition of abstract vacuity:

Definition 7. *A formula φ is ψ-vacuous in an abstract Kripke structure K_α iff it is ψ-vacuous in all concretizations of K_α. Formally,*

– $\forall K \in \mathcal{C}(K_\alpha) \cdot \forall K' \in \mathcal{B}(K) \cdot \forall x \in [S' \to \mathbf{2}] \cdot ||\varphi[\psi \leftarrow x]||^{K'}$, *or*
– $\forall K \in \mathcal{C}(K_\alpha) \cdot \forall K' \in \mathcal{B}(K) \cdot \forall x \in [S' \to \mathbf{2}] \cdot ||\neg\varphi[\psi \leftarrow x]||^{K'}$

Vacuity of abstract models. An exact abstraction K_α is always representable by a classical model, and its set of concretizations $\mathcal{C}(K_\alpha)$ is identical to the set $\mathcal{B}(K_\alpha)$ of structures bisimilar to K_α. Thus, vacuity given by Definition 4 also satisfies the conditions of Definition 7. As a consequence, an exact abstraction does not affect vacuity.

If K_α is not exact, the reduction from vacuity detection to 3-valued model-checking, as described in Section 4.2, provides us with a vacuity detection algorithm. Recall that K_x denotes a Kripke structure constructed from K by adding a new proposition x and letting it be maybe in every state of K.

Theorem 12. *A formula φ is ψ-vacuous in an abstract model K_α iff $\varphi[\psi \leftarrow x]$ evaluates to* true *or* false *in $(K_\alpha)_x$, under thorough semantics.*

In general, a non-exact abstraction can affect vacuity. A formula vacuous in a concrete Kripke structure K_c can be deemed non-vacuous in its abstraction. Thus, the result "non-vacuous" on an abstract system must be interpreted as "there is not enough information to decide whether the formula is vacuous".

Precision of vacuity detection. Recall that abstract model-checking is based on compositional semantics, while vacuity detection algorithm described above is based on thorough. Therefore, it is possible for vacuity detection to provide a more conclusive answer than abstract model-checking! A trivial example is a tautology, e.g., $AXp \vee EX\neg p$, which is true independently of the model and is vacuously true in p. However, in models where p is maybe, this formula is maybe under compositional semantics (used by abstract model-checking) but true under thorough. Thus, an answer that a formula is vacuously true in this case provides more information than results of abstract model-checking the same formula.

Approximating vacuity detection. Compositional semantics can also approximate vacuity detection in abstract models: if a formula $\varphi[\psi \leftarrow x]$ is true or false, then φ is ψ-vacuous. Of course, a maybe answer might need to be followed by a more expensive thorough semantics check.

6 Conclusion

In this paper, we give a uniform view of vacuity detection that is applicable to both branching- and linear-time temporal logics and yet enjoys all of the advantages of trace vacuity defined for LTL by Armoni et al [1]. This definition is *robust*, i.e., independent of logic embedding, and closed under bisimulation and thus independent of trivial changes to the model. Unfortunately, general vacuity detection is exponentially more expensive than model-checking. We show that for many useful fragments of temporal logic, vacuity detection is no more expensive than model-checking!

Furthermore, we identify a close connection between 3-valued model-checking and vacuity detection and note that the search for different definitions of vacuity corresponds to the search for different semantics for 3-valued model-checking. This connection suggests that there exists a natural relationship between the mature field of abstraction (not necessarily 3-valued) and the emerging field of vacuity detection. In particular, vacuity can be thought of as abstracting the formula instead of the model. In this paper, we started exploring this relationship by identifying what happens with vacuity under abstraction. Clearly, a much more thorough exploration is necessary; we leave it for future work.

References

1. R. Armoni, L. Fix, A. Flaisher, O. Grumberg, N. Piterman, A. Tiemeyer, and M. Vardi. "Enhanced Vacuity Detection in Linear Temporal Logic ". In *Proceedings of CAV'03*, pages 368–380, 2003.
2. T. Ball, A. Podelski, and S. Rajamani. "Boolean and Cartesian Abstraction for Model Checking C Programs". In *Proceedings of TACAS'01*, pages 268–283, 2001.
3. I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh. "Efficient Detection of Vacuity in ACTL Formulas". In *Proceedings of CAV'97*, 1997.
4. I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh. "Efficient Detection of Vacuity in Temporal Model Checking". *Formal Methods in System Design*, 18(2):141–163, March 2001.
5. G. Bruns and P. Godefroid. "Model Checking Partial State Spaces with 3-Valued Temporal Logics". In *Proceedings of CAV'99*, pages 274–287, 1999.
6. G. Bruns and P. Godefroid. "Generalized Model Checking: Reasoning about Partial State Spaces". In *Proceedings of CONCUR'00*, pages 168–182, 2000.
7. 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)*, volume 2404 of *LNCS*, July 2002.
8. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
9. T. French. "Decidability of Quantifed Propositional Branching Time Logics". In *Proceedings of AI 2001*, volume 2256 of *LNCS*, pages 165–176, 2001.
10. P. Godefroid, M. Huth, and R. Jagadeesan. "Abstraction-based Model Checking using Modal Transition Systems". In *Proceedings of CONCUR'01*, pages 426–440, 2001.
11. P. Godefroid and R. Jagadeesan. "Automatic Abstraction Using Generalized Model-Checking". In *Proceedings of CAV'02*, pages 137–150, July 2002.
12. P. Godefroid and R. Jagadeesan. "On the Expressiveness of 3-Valued Models". In *Proceedings of VMCAI'03*, pages 206–222, 2003.
13. S. Graf and H. Saidi. "Construction of Abstract State Graphs with PVS". In *Proceedings of CAV'97*, 1997.
14. A. Gurfinkel and M. Chechik. "Multi-Valued Model-Checking via Classical Model-Checking". In *Proceedings of CONCUR'03*, 2003.
15. A. Gurfinkel and M. Chechik. "How Vacuous Is Vacuous?". In *Proceedings of TACAS'04*, 2004. (To appear).
16. S. C. Kleene. *Introduction to Metamathematics*. New York: Van Nostrand, 1952.
17. O. Kupferman. "Augmenting Branching Temporal Logics with Existential Quantification over Atomic Propositions". *Journal of Logic and Computation*, 7:1–14, 1997.
18. O. Kupferman and M. Vardi. "Vacuity Detection in Temporal Model Checking". *STTT*, 4(2):224–233, 2003.
19. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.
20. R. Milner. "An Algebraic Definition of Simulation between Programs". In *Proceedings of IJCAI'91*, pages 481–489, 1971.

21. M. Purandare and F. Somenzi. "Vacuum Cleaning CTL Formulae". In *Proceedings of CAV'02*, pages 485–499, 2002.
22. S. Shoham and O. Grumberg. "A Game-Based Framework for CTL Counter-Examples and 3-Valued Abstraction-Refinement". In *Proceedings of CAV'03*, 2003.

A Proofs of Select Theorems

Theorem 2. *The complexity of detecting whether a formula φ is ψ-vacuous is EXPTIME-complete for CTL, and 2EXPTIME-complete for* CTL^*.

Proof:
In Section 3.1, we have shown that vacuity detection is reducible to the model-checking problem for quantified temporal logic with a single quantifier, under amorphous semantics. The membership follows from the fact that model-checking for this logic is reducible to model-checking it under tree semantics of Kupferman [17]. This yields a vacuity detection algorithm that is quadratic in the size of the model, and exponential in the size of the formula for CTL, and double-exponential for CTL^*.

Completeness follows from reducing satisfiability problem for CTL and CTL^* to QTL model-checking under amorphous semantics. The reduction is similar to the one used in the proof of Theorem 4.5 in [17], and is available in the extended version of the paper. $\square$

Theorem 3. *If a temporal logic formula φ is monotone in its subformula ψ, then the complexity of detecting whether φ is ψ-vacuous in a Kripke structure K is $2 \times MC(\varphi)$, where $MC(\varphi)$ is the complexity of model-checking φ in K.*

Proof:
We show that if φ is monotone in ψ, it is ψ-vacuous iff replacing ψ by both true and false does not affect the value of φ: $||\varphi[\psi \leftarrow \mathsf{true}]||^K = ||\varphi[\psi \leftarrow \mathsf{false}]||^K$. Since true and false are simply nullary functions from S to $\mathbf{2}$, the ($\Leftarrow$) direction follows trivially. Note that for any Kripke structure K' bisimilar to K, $\forall c \in \mathbf{2} \cdot ||\varphi[\psi \leftarrow c]||^K = ||\varphi[\psi \leftarrow c]||^{K'}$; thus, we only need to show that

$$(||\varphi[\psi \leftarrow \mathsf{true}]||^K = ||\varphi[\psi \leftarrow \mathsf{false}]||^K) \Rightarrow \forall x \in [S \to \mathbf{2}] \cdot ||\varphi[\psi \leftarrow x]||^K$$

which follows by monotonicity of φ. $\square$

Theorem 4. *The complexity of deciding whether a formula φ is monotone in a subformula ψ is EXPTIME-hard for CTL, and 2EXPTIME-hard for* CTL^*.

Proof:
We show that the validity problem for CTL is reducible to deciding monotonicity of a CTL formula. Consider the formula $(p \wedge q) \Rightarrow AX(p \wedge q)$ which is not monotone in $p \wedge q$. Let φ be an arbitrary CTL formula that does not contain atomic propositions p and q. Then, the formula $(p \wedge q \Rightarrow AX(p \wedge q)) \vee \varphi$ is monotone iff φ is valid. The proof for CTL^* is identical. $\square$

Theorem 5. *If φ is a CTL^* formula that is syntactically monotone in ψ, then φ is also monotone in ψ.*

Proof:
The proof follows from the monotonicity of CTL^* operators. $\square$

Theorem 6. *Let K be a Kripke structure, and K_x as described in Section 3.2. Then, if K' is $\{x\}$-bisimilar to K, then it is simulated by K_x.*

Proof:
Let $\rho \subseteq S \times S'$ be the $\{x\}$-bisimulation relation between K and K'. We claim that $\rho' = \{(\langle s, i\rangle, t) \mid \rho(s,t) \wedge I'(\langle s, i\rangle, x) = I_x(t,x)\}$ is a simulation between K_x and K'.

From the definition of ρ', we get $\rho(\langle s, i\rangle, t) \Leftrightarrow (I_x(\langle s, i\rangle) = I(t))$, thus, it satisfies the first condition of a simulation. The proof of the second condition is given below.

$$\begin{array}{l}
\quad \rho'(\langle s, i\rangle, t) \wedge R'(t, t') \\
\Rightarrow \text{(since } K' \text{ is } \{x\}\text{-bisimilar to } K) \\
\quad \exists s' \in S \cdot \rho(s', t') \wedge R(s, s') \\
\Rightarrow \text{(by definition of } K_x) \\
\quad \exists s' \in S \cdot \forall j \in \{0,1\} \cdot R_x(\langle s, i\rangle, \langle s', j\rangle) \wedge \rho(s', t') \\
\Rightarrow \text{(since } I'(t', x) \in \mathbf{2}) \\
\quad \exists s' \in S \cdot \exists j \in \{0,1\} \cdot R_x(\langle s, i\rangle, \langle s', j\rangle) \wedge \rho(s', t') \wedge I_x(\langle s', j\rangle) = I'(t') \\
\Rightarrow \text{(by definition of } \rho') \\
\quad \exists s' \in S \cdot \exists k \in \{0,1\} \cdot R_x(\langle s, i\rangle, \langle s', j\rangle) \wedge \rho'(\langle s', j\rangle, t')
\end{array}$$

Finally, if t is an initial state of K', then there exists an $i \in \{0,1\}$ such that $\rho(\langle s, i\rangle, t)$ holds, which establishes that ρ is a simulation between K_x and K'. □

Theorem 7. *Let φ be an ACTL^* formula, ψ be a subformula of φ, and K be a Kripke structure. If φ is known to be satisfied by K, then the complexity of detecting vacuity of φ in ψ is $2 \times MC(\varphi)$.*

Proof:
We show that a formula φ satisfied by K is ψ-vacuous iff the formula $\varphi[\psi \leftarrow x]$ is satisfied by K_x. Since K_x is $\{x\}$-bisimilar to K, the proof of ($\Rightarrow$) direction is trivial. For ($\Leftarrow$) direction, if $\varphi[\psi \leftarrow x]$ holds in K_x, then by Theorem 6 $\varphi[\psi \leftarrow x]$ it holds in every $\{x\}$-bisimulation of K. □

Theorem 8. *Let $\varphi[x, \psi]$ be an CTL^* formula, with a subformula x and a state subformula ψ, and let K be a Kripke structure. Then, φ is x-vacuous iff $\varphi[x, \psi \leftarrow Prop(\psi, K)]$ is x-vacuous.*

Proof:

$$\begin{array}{l}
\quad \forall K' \in \mathcal{B}(K) \cdot \forall x \in [S' \to \mathbf{2}] \cdot ||\varphi[x, \psi]||^{K'} \\
\Rightarrow \text{(by propositional substitution)} \\
\quad \forall K' \in \mathcal{B}(K) \cdot \forall x \in [S' \to \mathbf{2}] \cdot ||\varphi[x, \psi \leftarrow Prop(\psi, K')]||^{K'} \\
\Rightarrow \text{(since } K' \text{ is bisimilar to } K) \\
\quad \forall K' \in \mathcal{B}(K) \cdot \forall x \in [S' \to \mathbf{2}] \cdot ||\varphi[x, \psi \leftarrow Prop(\psi, K)]||^{K'}
\end{array}$$

□

Theorem 11. *Let K be a Kripke structure, and K_x be as described in the beginning of Section 4.2. Then, $K' \in \mathcal{C}(K_x)$ iff K' is $\{x\}$-bisimilar to K.*

Proof:
The proof follows trivially from the definitions of $\{x\}$-bisimilarity and refinement. □

Theorem 12. *A formula φ is ψ-vacuous in an abstract model K_α iff $\varphi[\psi \leftarrow x]$ does not evaluate to* maybe *in $(K_\alpha)_x$, under thorough semantics.*

Proof:

For an atomic proposition x of K, let K_{-x} denote a Kripke structure constructed from K by *removing* x from K; that is, $K = (K_x)_{-x}$. Note that both K and K_{-x} are defined over an identical set of states S. Furthermore, $(K_{-x})_x \preceq K$ under the identity relation $id = \{(s, s) \mid s \in S\}$.

Refinements of K_x are related to refinements of K: $K_x \preceq K' \Leftrightarrow K \preceq (K')_{-x}$ because x is maybe in every state of K_x, and the statespaces of K_x and K are identical. Thus, the set of concretizations of K_x is equivalent to a set obtained by: (a) concretizing K, (b) adding the atomic proposition x, and (c) taking the concretization of the result, i.e. $\mathcal{C}(K_x) = \mathcal{C}((\mathcal{C}(K))_x)$.

Finally, we prove the theorem.

$$
\begin{aligned}
& ||\varphi[\psi \leftarrow x]||_t^{(K_\alpha)_x} = \text{true} \\
\Leftrightarrow\ & \text{(by Definition 6)} \\
& \forall K \in \mathcal{C}((K_\alpha)_x) \cdot ||\varphi[\psi \leftarrow x]||^K = \text{true} \\
\Leftrightarrow\ & \text{(since } \mathcal{C}(K_x) = \mathcal{C}((\mathcal{C}(K))_x)\text{)} \\
& \forall K \in \mathcal{C}(\mathcal{C}((K_\alpha))_x) \cdot ||\varphi[\psi \leftarrow x]||^K = \text{true} \\
\Leftrightarrow\ & \text{(by Definition 6)} \\
& \forall K \in \mathcal{C}(K_\alpha) \cdot ||\varphi[\psi \leftarrow x]||_t^{K_x} = \text{true}
\end{aligned}
$$

The proof of the second case is similar. □