Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

LTL Generalized Model Checking Revisited

Patrice Godefroid[1], **Nir Piterman**[2⋆]

[1] Microsoft Research
[2] Imperial College London

Received: date / Revised version: date

Abstract. Given a 3-valued abstraction of a program (possibly generated using static program analysis and predicate abstraction) and a temporal logic formula, generalized model checking (GMC) checks whether there exists a concretization of that abstraction that satisfies the formula. In this paper, we revisit generalized model checking for linear time (LTL) properties. First, we show that LTL GMC is 2EXPTIME-complete in the size of the formula and polynomial in the model, where the degree of the polynomial depends on the formula, instead of EXPTIME-complete and quadratic as previously believed. The standard definition of GMC depends on a definition of concretization which is tailored for branching-time model checking. We then study a simpler *linear completeness preorder* for relating program abstractions. We show that LTL GMC with this weaker preorder is only EXPSPACE-complete in the size of the formula, and can be solved in linear time and logarithmic space in the size of the model. Finally, we identify classes of formulas for which the model complexity of standard GMC is reduced.

1 Introduction

Generalized model checking [BG00] is a way to improve precision when reasoning about partially defined systems. Such systems can be modeled as 3-valued Kripke structures where atomic propositions are either *true*, *false* or *unknown*, denoted by the third value $\perp$. Three-valued models are a natural representation of program abstractions generated automatically [GHJ01,GWC06] using static program analysis and predicate abstraction [GS97] for software model checking [BR01].

Given a 3-valued model M and a temporal-logic formula ϕ, the generalized model-checking (GMC) problem is to decide whether there exists a complete system M' that is consistent with M and that satisfies the formula ϕ. From a practical point of view, generalized model checking can sometimes [GH05,GC05] improve verification of program abstractions. From a theoretical point of view, studying GMC is arguably interesting in its own right since GMC generalizes both model checking (when the values of all propositions in the model are known) and satisfiability checking (when all proposition values are unknown), probably the two most studied problems related to temporal logic and verification.

In this paper, we revisit GMC for *linear-time temporal-logic* (LTL) formulas. We start by showing that LTL GMC is 2EXPTIME-complete in the size of the formula and polynomial in the model, where the degree of the polynomial depends on the formula, instead of EXPTIME-complete and quadratic as previously stated erroneously in [BG00]. The definition of GMC depends on the exact notion of abstraction, and is usually tailored for branching-time model checking [BG00]. We then study a simpler *linear completeness preorder* for relating program abstractions. We show that LTL GMC with this weaker preorder is only EXPSPACE-complete in the size of the formula, and can be solved in linear time and logarithmic space in the size of the model. Finally, we identify classes of formulas for which the model complexity of GMC defined with the standard branching-time completeness preorder is reduced.

Example. Consider the program P:

```
program P() {
  x,y = 1,0;
  x,y = 2*f(x),f(y);
  x,y = 1,0;
}
```

where `x` and `y` denote `int` variables, `f : int -> int` denotes some unknown function, and the notation "`x,y = 1,0`" means variables `x` and `y` are simultaneously assigned values `1` and `0`, respectively. Let ϕ_1 denote the LTL formula $Fq_y \wedge G(q_x \vee \neg q_y)$ with the two predicates q_x : "is x odd?" and q_y : "is y odd?", and where F means "eventually" while G means "always", and let ϕ_2 denote the LTL formula $Xq_y \wedge$

[⋆] Supported by the UK EPSRC project *Complete and Efficient Checks for Branching-Time Abstractions*

$G(q_x \vee \neg q_y)$, where X means“next” (see the next section for formal definitions).

Given such a program and knowing the predicate of interests q_x and q_y, predicate abstraction can be used to automatically generate the following 3-valued Kripke structure M (or “Boolean program” [BR01]) abstracting P [GHJ01]:

initial state s_0:	$q_x = true,$	$q_y = false$
next state s_1:	$q_x = false,$	$q_y = \perp$
next state s_2:	$q_x = true,$	$q_y = false$
loop forever in s_2		

As shown in [GJ02] and discussed later, model checking[1] ϕ_1 and ϕ_2 against M returns the value “unknown,” while generalized model checking can prove that no concretization of M can possibly satisfy either ϕ_1 or ϕ_2, i.e., no matter how function `f` is implemented.

Although $\phi_2 = Xq_y \wedge G(q_x \vee \neg q_y)$ is an LTL safety formula and hence is within the scope of predicate-abstraction-based software model checkers such as SLAM [BR01] or BLAST [HJMS02], these tools cannot prove that ϕ_2 does not hold regardless of the definition of function `f`: this result can only be obtained through generalized model checking. Instead, when confronted with such a program P, these tools would attempt to iteratively refine the abstraction M by analyzing the code of function `f` if it is available. This process is in general exponential in the size of the abstraction, since adding a single predicate in each iteration may double the size of the abstraction. Moreover, this process may not terminate. For the above abstraction M and formula ϕ_2, the expensive and unpredictable abstraction-refinement process can thus be avoided thanks to GMC. Although the worst-case complexity of GMC is expensive in the size of the (usually short) formula (but so is traditional LTL model checking which is already PSPACE-complete), GMC can always be done in time polynomial in the size of the model (linear or quadratic in many cases as shown later), in contrast with abstraction refinement which is typically exponential in the (usually large) model. □

2 Preliminaries

A *partial Kripke structure* (PKS for short) [BG99] is $M = \langle S, R, L, s^{in}\rangle$ where S is a nonempty set of states, $R \subseteq S \times S$ is a total image-finite transition relation (i.e., every state has a non-zero finite number of immediate successor states), $L : S \times AP \to \mathbf{3}$ is a labeling of states that associates a truth value in $\mathbf{3} = \{true, \perp, false\}$ to each atomic proposition in a finite set AP, and $s^{in} \in S$ is an initial state. For a state s and proposition p, we say that p is true in s if $L(s,p) = true$, it is false in s if $L(s,p) = false$, and it is unknown $\perp$ otherwise. A PKS is *complete* if the range of L is $\mathbf{2} = \{true, false\}$. We call a complete PKS a *Kripke Structure* or KS. When we want to stress that a PKS M is complete, we denote it by $\overline{M}$. Given a state s, we denote by $L(s)$ the function $\sigma : AP \to \mathbf{3}$ such that $\sigma(p) = L(s,p)$. We use the notations $\mathbf{3}^{AP} = \{\sigma : AP \to \mathbf{3}\}$ and $\mathbf{2}^{AP} = \{\sigma : AP \to \mathbf{2}\}$. For $s \in S$, we denote by (M, s) the PKS $\langle S, R, L, s\rangle$.

A *computation* of M is $s_0, s_1, \ldots$ such that $s_0 = s^{in}$ and forall $i \geq 0$ we have $(s_i, s_{i+1}) \in R$. A computation $\pi = s_0, s_1, \ldots$ induces a *trace* $L(\pi) = L(s_0)L(s_1)\cdots \in (\mathbf{3}^{AP})^\omega$. The set of computations of M is denoted $\mathcal{C}(M)$ and the set of traces of M is denoted $\mathcal{L}(M)$. In general, $\mathcal{L}(M) \subseteq (\mathbf{3}^{AP})^\omega$. Given a PKS $M = \langle S, R, L, s^{in}\rangle$, the *unwinding* of M into a tree is the PKS $M^+ = \langle S^+, R', L', s^{in}\rangle$, where S^+ is the set of nonempty sequences over S, $R' = \{(s_1 \cdots s_n, s_1 \cdots s_n \cdot s_{n+1}) \in (S^+ \times S^+) \mid (s_n, s_{n+1}) \in R\}$, and $L'(\pi \cdot s) = L(s)$. We restrict the set S^+ to the set of sequences reachable from s^{in}. If M is a Kripke structure then so is M^+.

To interpret temporal logic formulas on PKSs, we extend Kleene’s strong 3-valued propositional logic [Kle87]. Conjunction $\wedge$ in this logic is defined as the minimum Min of its arguments with respect to the *truth ordering* $\leq_T$ where $false \leq_T \perp \leq_T true$. We extend this function to sets in the obvious way, with $Min(\emptyset) = true$. Negation $\neg$ is defined using the function ‘$Comp$’ that maps $true$ to $false$, $false$ to $true$, and $\perp$ to $\perp$. Disjunction $\vee$ is defined as usual using De Morgan’s laws: $p \vee q = \neg(\neg p \wedge \neg q)$. Propositional modal logic (PML) is propositional logic extended with the modal operator AX (which is read “for all immediate successors”). Formulas of PML have the following abstract syntax:

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid AX\phi,$$

where p ranges over AP. The following 3-valued semantics generalizes the traditional 2-valued semantics for PML.

Definition 1. The value of a formula ϕ of 3-valued PML in a state s of a PKS $M = \langle S, R, L, s^{in}\rangle$, written $[(M,s) \models \phi]$, is defined inductively as follows:

$$\begin{aligned}
[(M,s) \models p] &= L(s,p) \\
[(M,s) \models \neg\phi] &= Comp([(M,s) \models \phi]) \\
[(M,s) \models \phi_1 \wedge \phi_2] &= Min(\{[(M,s) \models \phi_i] \mid i \in \{1,2\}\}) \\
[(M,s) \models AX\phi] &= Min(\{[(M,s') \models \phi] \mid (s,s') \in R\})
\end{aligned}$$

We write $[M \models \phi]$ for $[(M, s^{in}) \models \phi]$. This 3-valued logic can be used to define a preorder $\preceq$ on PKSs that reflects their degree of completeness. Let $\leq_I$ be the *information ordering* on truth values where $\perp$ is the least element and $true$ and $false$ are maximal uncomparable elements: $\perp \leq_I true, false$. For two PKS $M_i = \langle S_i, R_i, L_i, s_i^{in}\rangle$ with $i = 1, 2$ the *completeness preorder* is the greatest relation $\preceq \subseteq S_1 \times S_2$ such that $s_1 \preceq s_2$ implies all the following:

1. For every $p \in AP$, we have $L_1(s_1, p) \leq_I L_2(s_2, p)$.
2. For every $(s_1, s_1') \in R_1$, there exists $(s_2, s_2') \in R_2$ such that $s_1' \preceq s_2'$.
3. For every $(s_2, s_2') \in R_2$, there exists $(s_1, s_1') \in R_1$ such that $s_1' \preceq s_2'$.

We say that M_2 is *more complete* than M_1, denoted $M_1 \preceq M_2$, if $s_1^{in} \preceq s_2^{in}$. It can be shown that 3-valued PML logically characterizes the completeness preorder.

[1] In *model checking*, we mean normal 3-valued model checking in the sense of [BG99].

Theorem 1. [BG99] *Let M_1 and M_2 be partial Kripke structures, and let Φ be the set of all formulas of 3-valued PML. Then $M_1 \preceq M_2$ iff $(\forall \phi \in \Phi : [M_1 \models \phi] \leq_I [M_2 \models \phi])$.*

In other words, partial Kripke structures that are "more complete" with respect to $\preceq$ have more definite properties with respect to $\leq_I$, i.e., have more properties that can be established $true$ or $false$ by model checking. Moreover, any formula ϕ of 3-valued PML that evaluates to $true$ or $false$ on a partial Kripke structure has the same truth value when evaluated on any more complete structure.

2.1 Model Checking and Generalized Model Checking

The sets of LTL and CTL formulas are defined as follows.

$$\text{LTL } \varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid X\varphi \mid \varphi U \varphi$$

$$\text{CTL } \varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid AX\varphi \mid A\varphi U\varphi \mid E\varphi U\varphi$$

We assume familiarity with the semantics of LTL and CTL and with their model checking. As usual, we denote $true\ U\varphi$ by $F\varphi$, $\neg F \neg\varphi$ by $G\varphi$ and $\neg((\neg\psi)U(\neg\varphi \wedge \neg\psi))$ by $\varphi R \psi$. The above grammar includes a complete set of operators and other operators can be expressed in the usual way. Given a set of propositions AP and an LTL formula φ, the language of φ, denoted $L(\varphi)$ is the set of models of φ in $(\mathbf{2}^{AP})^\omega$. Formally, $L(\varphi) = \{w \in (\mathbf{2}^{AP})^\omega \mid w \models \varphi\}$. The 3-valued semantics of LTL and CTL path formulas extend Definition 1 as expected.

Precisely, given a 3-valued infinite word $w = a_0 a_1 a_2 \cdots \in (\mathbf{3}^{AP})^\omega$, the 3-valued semantics of the next LTL operator is defined as

$$[w \models X\varphi] = [w' \models \varphi]$$

with $w' = a_1 a_2 \cdots \in (\mathbf{3}^{AP})^\omega$, while the 3-valued semantics of the strong until LTL operator U is defined as

$$[w \models \varphi_1 U \varphi_2] =$$

$$Max(\{Min(\{[a_i \models \varphi_1] \mid i < k\} \cup \{[a_k \models \varphi_2]\}) \mid k \geq 0\})$$

For a partial Kripke structure M and a CTL formula ϕ, we denote the value of ϕ at state s by $[(M, s) \models \phi] \in \mathbf{3}^{AP}$. Let $Path(s)$ denote all 3-valued infinite paths starting at s. The 3-valued semantics of the CTL operators AU and EU are then defined as

$$[(M, s) \models A\varphi_1 U\varphi_2] = Min(\{[w \models \varphi_1\, U\varphi_2] \mid w \in Path(s)\})$$

and as

$$[(M, s) \models E\varphi_1\, U\varphi_2] = Max(\{[w \models \varphi_1\, U\varphi_2] \mid w \in Path(s)\})$$

For the initial state s^{in} of M we denote $[(M, s^{in}) \models \phi]$ by $[M \models \phi]$. If M is a Kripke structure we simply write $M, s \models \varphi$ for $[(M, s) \models \varphi] = true$ and $M, s \not\models \varphi$ for $[(M, s) \models \varphi] = false$. For a Kripke structure $\overline{M}$ and an LTL formula φ, we say that $\overline{M}$ satisfies φ, denoted $\overline{M} \models \varphi$ if $\mathcal{L}(\overline{M}) \subseteq L(\varphi)$.

In practice, the size of the Kripke structure $\overline{M}$ can be prohibitively expensive or even infinite. Instead, a smaller (finite) *abstraction* M' can be used: if M' is generated in such a way that $M' \preceq \overline{M}$, then all the properties ϕ that can be proved ($true$) or disproved ($false$) on M' will also hold on $\overline{M}$, by Theorem 1. With static program analysis and predicate abstraction, generating such abstractions with respect to the completeness preorder $\preceq$ can be done at the same computational cost as computing standard abstractions that merely simulate (over-approximate) the concrete system $\overline{M}$ [GHJ01]. Moreover, 3-valued model checking can itself be done at the same computational cost as regular 2-valued model checking [BG00].

In some cases, precisely characterized in [GH05] and also independently studied in [GC05], all the completions of an abstraction M agree on the satisfaction of a formula φ, yet 3-valued model checking is not accurate enough to identify this and still returns $\perp$. For instance, this is the case for the formula $p \vee \neg p$ if p is $\perp$. This observation suggests a more precise version of 3-valued model checking [BG00]: the value of a formula φ in a PKS M should be unknown only if some completions of M satisfy φ *and* some completions of M falsify φ [BG00]. We denote the value of φ on M according to this *thorough semantics* by $[M \models \varphi]_t \in \mathbf{3}$.

Generalized model checking (GMC) can determine the value of $[M \models \varphi]_t$ [BG00]. Given a PKS M and a formula φ, the GMC problem for M and φ is to determine whether there exists a Kripke structure M' that completes M and satisfies φ. Formally, we have the following.

$$M \models_{\preceq} \varphi \text{ iff there exists } \overline{M'} \succeq M \text{ such that } \overline{M'} \models \varphi$$

The value $[M \models \varphi]_t$ can be evaluated with two GMC questions. First, we check whether $M \models_{\preceq} \varphi$. If the answer is no, then all completions of M do not satisfy φ and $[M \models \varphi]_t = false$. If the answer is yes, we next check whether $M \models_{\preceq} \neg\varphi$. If that answer is no, then we know that all completions of M satisfy φ and $[M \models \varphi]_t = true$. Otherwise, $[M \models \varphi]_t = \perp$.

It can be shown that 3-valued model checking is sound with respect to the thorough semantics.

Theorem 2. [BG00] *Let M be a PKS and φ an LTL or CTL formula.*
1. $[M \models \varphi] = true$ implies $[M \models \varphi]_t = true$.
2. $[M \models \varphi] = false$ implies $[M \models \varphi]_t = false$.

In this paper we revisit LTL generalized model checking and show that its complexity is greater than what was previously believed. We also consider specifications (both in LTL and CTL) for which the model complexity of generalized model checking is simpler than the general case.

2.2 Automata over Infinite Words

We assume familiarity with the basic notions of alternating automata on infinite words, cf. [GTW02]. We also refer to tree automata, however, we do not define them formally.

For an alphabet Σ, the set Σ^* is the set of finite sequences of elements from Σ. For $x \in \Sigma^*$, we denote the length of x by $|x|$. Given an alphabet Σ and a set D of directions, a *Σ-labeled D-tree* is a pair $\langle T, \tau\rangle$, where $T \subseteq D^*$ is a tree over D and $\tau : T \to \Sigma$ maps each node of T to a letter in Σ. A *path* π of a tree T is a set $\pi \subseteq T$ such that $\epsilon \in \pi$ and for every $x \in \pi$ either x is a leaf in T or there exists a unique $\gamma \in D$ such that $x \cdot \gamma \in \pi$. For $\pi = \gamma_1 \cdot \gamma_2 \cdots$, we write $\tau(\pi)$ for $\tau(\epsilon) \cdot \tau(\gamma_1) \cdot \tau(\gamma_1\gamma_2) \cdots$.

For a finite set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas *true* and *false*. For a set $Y \subseteq X$ and a formula $\theta \in \mathcal{B}^+(X)$, we say that Y *satisfies* θ iff assigning *true* to elements in Y and assigning *false* to elements in $X \setminus Y$ makes θ true. An alternating word automaton is $A = \langle \Sigma, Q, q_{in}, \delta, \alpha\rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function, $q_{in} \in Q$ is an initial state, and α specifies the acceptance condition. A run of A on $w = \sigma_0\sigma_1 \cdots$ is a Q-labeled D-tree, $\langle T, \tau\rangle$, where $\tau(\epsilon) = q_{in}$ and, for every $x \in T$, we have $\{\tau(x \cdot \gamma_1), \ldots, \tau(x \cdot \gamma_k)\} \models \delta(\tau(x), \sigma_{|x|})$ where $\{x \cdot \gamma_1, \ldots, x \cdot \gamma_k\}$ is the set of children of x. A run of A is accepting if all its infinite paths satisfy the acceptance condition. For a path π, we denote the set of automaton states visited infinitely often along this path by $inf(\pi)$, i.e., $inf(\pi) = \{q \mid q \text{ appears infinitely often in } \tau(\pi)\}$. We consider the following three acceptance conditions:

- A path π satisfies a *Büchi* condition $\alpha \subseteq Q$ iff $inf(\pi) \cap \alpha \neq \emptyset$.
- A path π satisfies a *co-Büchi* condition $\alpha \subseteq Q$ iff $inf(\pi) \cap \alpha = \emptyset$.
- A path π satisfies a *parity* condition $\alpha = \langle F_0, \ldots, F_k\rangle$ where $F_0, \ldots F_k$ form a partition of Q iff for some even i we have $inf(\pi) \cap F_i \neq \emptyset$ and forall $i' < i$ we have $inf(\pi) \cap F_{i'} = \emptyset$. We call k the number of *priorities* of α.

For the three conditions, an automaton accepts a word iff there exists a run that accepts it. We denote by $\mathcal{L}(\mathcal{A})$ the set of all Σ-words that A accepts.

Below we discuss some special cases of alternating automata. The alternating automaton A is *nondeterministic* if for all the formulas that appear in δ are disjunctions over the states Q. The automaton A is *deterministic* if all formulas that appear in δ are states from Q. For a nondeterministic automaton we write $\delta : Q \times \Sigma \to 2^Q$ and for a deterministic automaton we write $\delta : Q \times \Sigma \to Q$.

We denote each of the different types of automata by an acronym in $\{D, N, A\} \times \{W, B, C, P\} \times \{W, T\}$, where the first letter describes the branching mode of the automaton (deterministic, nondeterministic, or alternating), the second letter describes the acceptance condition (Weak,[2] Büchi, co-Büchi, or parity), and the third letter describes the object over which the automaton runs (words or trees). For example, an ABW is an alternating Büchi word automata and a DPW is a deterministic parity word automata.

We state the following well known results about automata and their relation to LTL.

Theorem 3. *For every LTL formula φ of length n there exist an NBW N_φ with $2^{O(n)}$ states* [VW94] *and a DPW D_φ with $2^{2^{O(n\ log n)}}$ states and $2^{O(n)}$ priorities* [Saf88,Pit07] *such that $L(\varphi) = L(N_\varphi) = L(D_\varphi)$.*

Theorem 4. [Jur00] *Given an APW A over a 1-letter alphabet with n states and k priorities, we can decide whether $L(A) = \emptyset$ in time proportional to $n^{O(k)}$.*

Theorem 5. [SVW87] *Given two NBW N_1, N_2 we can decide whether $L(N_1) \subseteq L(N_2)$ in space logarithmic in N_1 and polynomial in N_2.*

3 LTL Generalized Model Checking

We show that, contrary to previous beliefs, GMC with respect to linear time logic is 2EXPTIME-complete. Our upper bound combines a DPW for the LTL property with the PKS to get an APW over a 1-letter alphabet. The APW is not empty iff the GMC problem holds. For the lower bound, we show a reduction from LTL realizability to generalized model checking. LTL realizability is 2EXPTIME-hard [PR89] establishing 2EXPTIME-hardness of generalized model checking. The two together establish 2EXPTIME-completeness of generalized model checking for LTL.

Theorem 6. *LTL generalized model checking $M \models_{\preceq} \varphi$ can be solved in polynomial time in the size of M and double exponential time in the size of φ.*

Proof. Consider an LTL formula φ. Let $|\varphi| = n$. According to Theorem 3 there exists a DPW D_φ with $2^{2^{O(n\ log\ n)}}$ states and $2^{O(n)}$ priorities such that $L(\varphi) = L(D_\varphi)$.

Let $D_\varphi = \langle \mathbf{2}^{AP}, T, t_0, \rho, \alpha\rangle$ and $M = \langle S, R, L, s^{in}\rangle$. Consider the following APW A over a 1-letter alphabet that is obtained from the combination of M and D_φ. We define $A = \langle \{a\}, T \times S, (t_0, s^{in}), \eta, \alpha'\rangle$ such that

$$\eta((t,s),a) = \bigvee_{\overline{\sigma} \succeq L(s)} \ \bigwedge_{(s,s')\in R} (\rho(t,\overline{\sigma}), s')$$

and $\alpha' = \langle F_0', \ldots, F_k'\rangle$ is obtained from $\alpha = \langle F_0, \ldots, F_k\rangle$ by setting $F_j' = F_j \times S$.

Lemma 1. *A accepts a^ω iff $M \models_{\preceq} \varphi$.*

Proof. Suppose that A accepts a^ω. Let $\langle T, \tau\rangle$ be an accepting run of A on a^ω. Consider the following Kripke structure $\langle T, R', L\rangle$ where $R'(x, xa)$ for every $x, xa \in T$ and $L : T \to \mathbf{2}^{AP}$ is defined as follows. Consider a node x where $\tau(x) = (t, s)$ and $\{xa_1, \ldots, xa_k\}$ is the set of children of x. We define $L(x)$ as the function $\sigma : AP \to \{true, false\}$ such that $\{\tau(xa_1), \ldots, \tau(xa_k)\} \models \bigwedge_{(s,s')\in R}(\rho(t, \sigma), s')$. By definition of a run of A such a function exists. It is easy to see that $\langle T, R', L\rangle$ is a Kripke structure that is more complete

[2] We delay the definition of weak automata to Section 5.

than M. Moreover, for any path π in $\langle T, R', L\rangle$, the projection of τ on the states of D_φ for this path defines an accepting run of D_φ and implies that $L(\pi) \models \varphi$.

Conversely, suppose that there exists a Kripke structure $K = \langle S, R, L, s_0\rangle$ such that $M \preceq K$ and $K \models \varphi$. Consider the unwinding $K^+ = \langle S^+, R', L', s_0\rangle$ of K. We construct a labeling $\tau : S^+ \to T \times S$ such that for every node $x = s_0 s_1 \cdots s_n$ we have $\tau(x)$ associates x to a state s of M such that $s \preceq s_n$ and to a state d of D_φ such that $d = \rho(d_0, L(s_0 \cdots s_{n-1}))$. Note that d is unique for a given sequence $s_0 \cdots s_{n-1}$ as the automaton D_φ is *deterministic*. Initially, $\tau(s_0) = (t_0, s^{in})$. Obviously, $s^{in} \preceq s_0$. Consider a node $x = s_0 s_1 \cdots s_n$ such that $\tau(x) = (t, s)$. Let $x s_{n+1}$ be a child of x. Then $\tau(x s_{n+1}) = (t', s')$ where $t' = \rho(t, L(s_n))$ and as $s \preceq s_n$ there exists a successor s' of s such that $s' \preceq s_{n+1}$. It is easy to see that $\langle S^+, \tau\rangle$ is a run tree of A on a^ω. By assumption every trace of K is accepted by D_φ. It follows that every infinite path in $\langle S^+, \tau\rangle$ is labeled by an accepting run of D_φ and that $\langle S^+, \tau\rangle$ is accepting.

According to Theorem 4 the emptiness of A can be determined in time proportional to $(2^{2^{O(n\ log\ n)}})^{2^{O(n)}} = 2^{2^{O(n\ log\ n)}}$. □

Note that, if D_φ was nondeterministic in the previous proof, it could not precisely track simultaneously different matching states s such that $s \preceq s_n$ in the proof, and therefore $M \models_\preceq \varphi$ would not necessarily imply that A accepts a^ω. This is in essence the error in the proof of Theorem 25 of [BG00], which led to the overly optimistic EXPTIME upper-bound.

We now proceed to the lower bound. We start with a definition of LTL realizability. Consider a set of propositions $AP = I \cup O$ of input and output signals, respectively. Let L be a language of infinite words over alphabet $\mathbf{2}^{AP}$. The *realizability problem* for L is to decide whether there exists a strategy $f : (2^I)^+ \to 2^O$ such that all the computations generated by f are in L. A *computation* $\pi = (i_0, o_0), (i_1, o_1), \ldots$ is generated by f if for all $j \geq 0$ we have $o_j = f(i_0 i_1 \cdots i_j)$. The realizability problem for an LTL formula φ is the realizability problem for $L(\varphi)$.

Theorem 7. [PR89] *The realizability problem for an LTL formula φ is 2EXPTIME-hard in the size of φ.*

Theorem 8. *LTL Generalized model checking $M \models_\preceq \varphi$ is 2EXPTIME-hard in the size of φ.*

Proof. We show how to solve realizability of an LTL formula using the generalized model checking problem. The idea behind the reduction is that the PKS includes determined values of the inputs and undetermined values of the outputs. The branching of the PKS forces all possible assignments to inputs as possible successors of every state. Thus, every completion of the PKS associates an assignment to the outputs with every possible assignment to inputs and is in essence a strategy. If the completion satisfies the LTL formula, then so does the strategy. The PKS has 2^I different states, each labeled by the appropriate assignment to the input variables and with transitions between every two possible states. Formally, we have the following.

Consider the PKS $M = \langle S, R, L, s^{in}\rangle$ with $S = 2^I$, $R = S \times S$, and $s^{in} \in S$ is chosen arbitrarily. For every proposition $i \in I$ we have $L(s, i) = \mathit{true}$ if $i \in s$ and $L(s, i) = \mathit{false}$ otherwise. For every proposition $o \in O$ we have $L(s, o) = \bot$. Consider the problem of generalized model checking of $X\varphi$ on M.

Suppose that $M \models_\preceq X\varphi$. There exists $\overline{M'} = \langle S', R', L', s'_0\rangle$ such that $M \preceq \overline{M'}$ and $\overline{M'} \models X\varphi$. For every state $t \in S'$ and every assignment $\sigma : I \to \{\mathit{true}, \mathit{false}\}$ there exists a state $t' \in S'$ such that $(t, t') \in R'$ and $L(t')$ agrees with σ on all the propositions in I. For $\sigma \in 2^I$ let $\sigma(t)$ denote this state t'. It follows that every sequence $\sigma_1 \cdots \sigma_k \in (2^I)^+$ induces a sequence $s'_1, \ldots, s'_k$ such that $s'_1 = \sigma_1(s'_0)$ and $s'_{i+1} = \sigma_{i+1}(s'_i)$. Denote $s'_1, \ldots, s'_k$ by $\sigma_1 \cdots \sigma_k(s'_0)$ and let $O(s'_1, \ldots, s'_k)$ denote the projection of $L'(s'_k)$ on the propositions in O. Consider the strategy $f : (2^I)^+ \to 2^O$ where for $w \in (2^I)^+$ we set $f(w) = O(w(s'_0))$. Consider a computation $\pi = (i_1, o_1), \ldots$ generated by f. By construction $(i_0, o_0), (i_1, o_1), \ldots$ is a computation of M where $(i_0, o_0) = L'(s'_0)$. As $\overline{M'} \models X\varphi$ we conclude that $\pi \models \varphi$.

Suppose that φ is realizable. There exists a strategy $f : (2^I)^+ \to 2^O$ such that every computation induced by f satisfies φ. Consider the tree $\langle (2^I)^*, L'\rangle$ where $L'(w \cdot i) = i \cup f(w)$ and $L'(\epsilon)$ agrees with $L(s^{in})$ on the propositions in I. It is simple to see that $\langle (2^I)^*, L'\rangle$ completes M and satisfies $X\varphi$.

This construction results in an exponential PKS. In Appendix A we show how to represent the same concept with a PKS with constant size and $|O| + 2$ propositions. □

4 Linear Completeness Preorder

The completeness preorder $\preceq$ used to define the generalized model checking relation $\models_\preceq$ is stronger than necessary for reasoning only about the linear behaviors of partial Kripke structures. Indeed, the completeness preorder reduces to a bisimulation relation in the case of complete Kripke structures, and Kripke structures that satisfy the same LTL formulas are not necessarily bisimilar.

In this section, we study a simpler *linear completeness preorder* $\preceq_L$, first suggested in [BG00], that relates partial Kripke structures using only their sets of (3-valued) traces. Then we show that generalized model checking $\models_{\preceq_L}$ defined with respect to this linear preorder is "only" EXPSPACE-complete.

Given any two infinite 3-valued traces $w{=}L(s_0)L(s_1)\cdots$ and $w'{=}L(s'_0)L(s'_1)\cdots$ in $(\mathbf{3}^{AP})^\omega$, we write $w \leq_I w'$ if $\forall i \geq 0 : \forall p \in AP : L(s_i, p) \leq_I L(s'_i, p)$.

Definition 2. For two PKS $M_i = \langle S_i, R_i, L_i, s_i^{in}\rangle$ with $i = 1, 2$, the *linear completeness preorder* $\preceq_L$ is the greatest relation $\preceq_L \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in \preceq_L$ implies all the following.

1. For every $w \in \mathcal{L}(M_1, s_1)$ there exists $w' \in \mathcal{L}(M_2, s_2)$ such that $w \leq_I w'$.

2. For every $w' \in \mathcal{L}(M_2, s_2)$ there exists $w \in \mathcal{L}(M_1, s_1)$ such that $w \leq_I w'$.

It is easy to show that 3-valued LTL logically characterizes the linear completeness preorder.

Theorem 9. *For any two PKS M_1 and M_2, we have $M_1 \preceq_L M_2$ iff for every LTL formula φ we have $[M_1 \models \varphi] \leq_I [M_2 \models \varphi]$.*

Proof. Assume $M_1 \preceq_L M_2$ and consider any LTL formula φ. If $[M_1 \models \varphi] = \bot$, we always have $[M_1 \models \varphi] \leq_I [M_2 \models \varphi]$.

If $[M_1 \models \varphi] = \mathit{true}$, then for all $w \in \mathcal{L}(M_1)$, $[w \models \varphi] = \mathit{true}$. By point 2 of Definition 2, for every $w' \in \mathcal{L}(M_2)$ there exists $w \in \mathcal{L}(M_1)$ such that $w \leq_I w'$. But since $\forall w \in \mathcal{L}(M_1) : [w \models \varphi] = \mathit{true}$, we have $\forall w' \in \mathcal{L}(M_2) : [w' \models \varphi] = \mathit{true}$, and hence $[M_2 \models \varphi] = \mathit{true}$.

If $[M_1 \models \varphi] = \mathit{false}$, then $\exists w \in \mathcal{L}(M_1) : [w \models \varphi] = \mathit{false}$. By point 1 of Definition 2, we have $\exists w' \in \mathcal{L}(M_2) : w \leq_I w'$ and hence $[w' \models \varphi] = \mathit{false}$. Thus $[M_2 \models \varphi] = \mathit{false}$, and the first direction of the theorem holds.

Conversely, let $s_1 \sqsubseteq s_2$ denote $\forall \varphi \in LTL : [(M_1, s_1) \models \varphi] \leq_I [(M_2, s_2) \models \varphi]$. Assume that $s_1 \sqsubseteq s_2$ but that $s_1 \not\preceq_L s_2$: thus, either point 1 or 2 of Definition 2 is violated.

Assume point 1 is violated: $\exists w \in \mathcal{L}(M_1, s_1) : \forall w' \in \mathcal{L}(M_2, s_2) : w \not\leq_I w'$. Let $w = s_1^0 s_1^1 s_1^2 \cdots$ with $s_1^0 = s_1$. Let $S_2^0 = \{s_2\}$ and for $k > 0$, let $S_2^k = \{s \in S_2 \mid s' \in S_2^{k-1} \wedge (s', s) \in R_2 \wedge (\forall p \in AP : L_1(s_1^k, p) \leq_I L_2(s, p))\}$. Since $\forall w' \in \mathcal{L}(M_2, s_2) : w \not\leq_I w'$, then there must exist a value of k such that $S_2^k = \emptyset$. In other words, the corresponding s_1^k in M_1 denote the first state in M_1 reachable from s_1 along w whose label cannot be “matched” (according to the previous formal definition) by any state of M_2 (hence also reachable in k steps from s_2). By abusing notation, let $S_2^k = \{s \in S_2 \mid s' \in S_2^{k-1} \wedge (s', s) \in R_2\}$ (by construction, we know $S_2^{k-1} \neq \emptyset$ and since every state has at least one successor state, S_2^k is nonempty as well). Thus, for each state $s \in S_2^k$, there exists a proposition $p \in AP$ such that $L_1(s_1^k, p) \not\leq_I L_2(s, p)$. Let $\varphi(s) = p$ if $L_1(s_1^k, p) = \mathit{false}$ and let $\varphi(s) = \neg p$ otherwise (i.e., when $L_1(s_1^k, p) = \mathit{true}$; if $L_1(s_1^k, p) = \bot$, then trivially $L_1(s_1^k, p) \leq_I L_2(s, p)$). Consider the LTL formula

$$\psi = (\textstyle\bigwedge_{i<k} (X^i (\varphi_{L(s_1^i)}))) \Rightarrow X^k \textstyle\bigvee_{s \in S_2^k} \varphi(s),$$
$$\text{where } \varphi_{L(s_1^i)} = \bigwedge_{L(s_1^i, p) = \mathit{true}} p \wedge \bigwedge_{L(s_1^i, p) = \mathit{false}} \neg p$$

We have $[(M_1, s_1) \models \psi] = \mathit{false}$ (as we know $[w \models \psi] = \mathit{false}$) while $[(M_2, s_2) \models \psi] \neq \mathit{false}$ (since the antecedent of the logical implication is true exactly for finite paths leading to states in S_2^{k-1} and the consequent is either true or $\bot$ for all states in S_2^k). A contradiction with $s_1 \sqsubseteq s_2$.

Assume point 2 is violated: $\exists w' \in \mathcal{L}(M_2, s_2) : \forall w \in \mathcal{L}(M_1, s_1) : w \not\leq_I w'$. Using the same line of reasoning as in the previous case, let s_2^k denote the first state in M_2 reachable from s_2 along w' whose label cannot be matched by any state in S_1^k of M_1 as defined above. Thus, for each state $s \in S_1^k$, there exists a proposition $p \in AP$ such that $L_1(s, p) \not\leq_I L_2(s_2^k, p)$. Let $\varphi(s) = p$ if $L_1(s, p) = \mathit{true}$ and let $\varphi(s) = \neg p$ otherwise. Consider the LTL formula

$$\psi = (\textstyle\bigwedge_{i<k} (X^i (\varphi_{L(s_2^i)}))) \Rightarrow X^k \textstyle\bigvee_{s \in S_1^k} \varphi(s),$$
$$\text{where } \varphi_{L(s_2^i)} = \bigwedge_{L(s_2^i, p) = \mathit{true}} p \wedge \bigwedge_{L(s_2^i, p) = \mathit{false}} \neg p \wedge \bigwedge_{L(s_2^i, p) = \bot} (p \wedge \neg p)$$

We have $[(M_1, s_1) \models \psi] = \mathit{true}$ (since the antecedent of the logical implication is either true or $\bot$ exactly for the finite paths leading to states in S_1^{k-1} and the consequent is true for all states in S_1^k) while $[(M_2, s_2) \models \psi] \neq \mathit{true}$ (since $[w' \models \psi] \neq \mathit{true}$). A contradiction with $s_1 \sqsubseteq s_2$. □

Given a PKS M and an LTL formula φ, generalized model checking with respect to the linear completeness preorder $\preceq_L$ means checking whether every 3-valued trace of M can be completed to a 2-valued trace that satisfies φ. Formally, we have the following.

$$M \models_{\preceq_L} \varphi \text{ iff } \forall w \in \mathcal{L}(M) :$$
$$\exists \text{ a complete } w' \text{ such that } w \leq_I w' \text{ and } w' \models \varphi$$

As observed in [GJ02], computing the value of $[M \models \varphi]_t$ for an LTL formula φ can be reduced to one normal (2-valued) model checking problem and one generalized model checking problem, regardless of which completeness preorder is used. One can start by checking whether there exists a completion w' of any trace w in M such that $w' \not\models \varphi$. To do this, one can build a Kripke structure M^c that guesses all possible completions of labelings of states of M and thus accepts all the possible completions of traces of M. Then, one checks whether $M^c \models \varphi$ using traditional 2-valued LTL model checking, which is a PSPACE-complete problem. If $M^c \models \varphi$, all possible completions of M satisfy φ, which means $[M \models \varphi]_t = \mathit{true}$ and we stop. Otherwise, one needs to solve a second, more expensive generalized model checking problem to determine whether there exists some completion M' of M whose traces all satisfy φ.

If one considers the completeness preorder $\preceq$, checking for such a completion $M' \succeq M$ such that $M' \models \varphi$, i.e., computing $M \models_{\preceq} \varphi$, is 2EXPTIME-complete as shown in the previous section. However, if one considers instead the *linear* completeness preorder $\preceq_L$, we now show that computing $M \models_{\preceq_L} \varphi$ is only EXPSPACE-complete.

Theorem 10. *LTL generalized model checking $M \models_{\preceq_L} \varphi$ with respect to the linear completeness preorder $\preceq_L$ can be solved in space logarithmic in the size of M and exponential in the size of φ.*

Proof. Consider an LTL formula φ. According to Theorem 3 there exists an NBW $N_\varphi = \langle \mathbf{2}^{AP}, Q, q_0, \rho, F \rangle$ where $|Q| = 2^{O(|\varphi|)}$ such that $L(N_\varphi) = L(\varphi)$.

We modify the NBW above to an NBW over the alphabet $\mathbf{3}^{AP}$ that accepts partial traces that have a completion in $L(N_\varphi)$. Formally, we have the following.

We denote letters in $\mathbf{2}^{AP}$ by $\overline{\sigma}$ and letters in $\mathbf{3}^{AP}$ by τ. Let N' be the automaton obtained from N_φ by guessing a completion of the read letter. Formally, $N' = \langle \mathbf{3}^{AP}, Q, q_0, \rho', F\rangle$ where

$$\rho'(s,\tau) = \bigvee_{\overline{\sigma}\succeq\tau} \rho(s,\overline{\sigma})$$

Now, all that we have to check is whether $L(M) \subseteq L(N')$. From Theorem 5, we know that this problem can be solved in space logarithmic in M and polynomial in N'. As N' is exponential in φ, the upper bound follows. □

We now show that using this definition of GMC we can solve an EXPSPACE-hard tiling problem [vEB97]. In tiling problems we get a finite set of different types of tiles and we have to tile a floor of a given dimension. We may use as many tiles as we want from every given type, however, there are rules that tell us which tiles are allowed to be next to each other according to vertical and horizontal rules. There are many different flavors of tiling problems with different complexities. Here we introduce the EXPSPACE version of the tiling problem. In order to prove the lower bound, we build a PKS M whose traces are all the possible arrangements of tiles. A trace has a completion that satisfies our LTL formula φ if the arrangement of tiles is not valid, i.e., it violates one of the tiling rules. That is, $M \models_{\preceq_L} \varphi$ iff all possible arrangements of tiles are not valid, i.e., the tiling problem does not have a solution.

A *tiling problem* is $\langle T, H, V, s, t, n\rangle$, where T is a finite set of tiles, $H, V \subseteq T \times T$ are horizontal and vertical consistency rules, $s, t \in T$ are initial and final tiles, and n is a number (in unary). The decision problem is whether there exists a number m and a function $f : [2^n] \times [m] \to T$ such that $f(1,1) = s$, $f(2^n, m) = t$, and forall i, j we have $(f(i,j), f(i+1,j)) \in H$ and $(f(i,j), f(i,j+1)) \in V$. That is, arrange the tiles in a 2^n times m rectangle such that s is in the bottom left corner, t in the top right corner, and all neighbors (vertical/horizontal) satisfy the horizontal and vertical consistency rules. This problem is EXPSPACE-complete [vEB97].

Theorem 11. *LTL generalized model checking $M \models_{\preceq_L}$ with respect to the linear completeness preorder $\preceq_L$ is EXPSPACE-hard in the size of φ.*

Proof. We start by representing the rectangular arrangement of tiles by a linear sequence of tiles. An (infinite) linear sequence of tiles represents a valid tiling if it starts with s, has t in location $m2^n$ for some m, every adjacent locations (except multiples of 2^n and their successors) satisfy H, and every two locations whose distance is 2^n satisfy V.

We construct a simple system that produces all possible sequences of tiles. The partial propositions are going to number every tile in the sequence with a number in $[0..(2^n - 1)]$. The LTL formula checks two things. First, that the truth assignments to partial propositional variables behave like a counter (it is always possible to complete the values of these propositions in this way). Second, that every possible sequence of tiles contains one of the following problems: either (a) it does not start in s, or (b) all locations that are multiples of 2^n are not t, or (c) the horizontal rule is violated before t appears in a 2^n-multiple location, or (d) the vertical rule is violated before t appears in a 2^n-multiple location. If one of these problems occurs, then the tiling is not valid. If all possible arrangements of tiles are not valid, then the tiling problem does not have a solution.

Suppose that $|T| = I = 2^k$, then the system is the I-clique. We have k propositions $\{p_1, \ldots, p_k\}$ and every state encodes one possible tile. The initial state is the state that corresponds to s. We have n propositions $\{q_1, \ldots, q_n\}$ that encode the location of the tile modulo 2^n. Finally, we have two more propositions $\{a, b\}$ that are used to identify the location of a violation of the vertical rule. In all states of the system the value of $\{a, b, q_1, \ldots, q_n\}$ is unknown. Formally, let $M = \langle S, R, L, s\rangle$ where $S = 2^{\{p_1,\ldots,p_k\}}$, $R = S \times S$, $L(t, p_i) = \mathit{true}$ iff $p_i \in t$, $L(t, q) = \bot$ for $q \in \{a, b, q_1, \ldots, q_n\}$, and by abuse of notation s is the state that corresponds to the initial tile s.

The LTL formula is $\varphi = \varphi_1 \wedge \varphi_2$, where φ_1 and φ_2 are defined below.

1. Counter consistency – φ_1 is conjunction of the following formulas.
   (a) The counter starts at 0:
   $$\bigwedge_{j=1}^{n} \neg q_j$$
   (b) The counter respects normal counting:
   $$\begin{array}{l} G\,(q_1 \leftrightarrow X\neg q_1) \hfill \wedge \\ G\left(\bigwedge_{j=2}^{n}\left(\big(\bigwedge_{j'<j} q_{j'}\big) \rightarrow (q_j \leftrightarrow X\neg q_j)\right)\right) \wedge \\ G\left(\bigwedge_{j=2}^{n}\left(\big(\bigvee_{j'<j} \neg q_{j'}\big) \rightarrow (q_j \leftrightarrow X q_j)\right)\right) \end{array}$$
2. The formula φ_2 describes the possible problems in the sequence of tiles. Let ϕ_e stand for $(\bigwedge_{j=1}^{n} q_j)$, i.e., the counter is $2^n - 1$ and let let ϕ_t stand for $(\phi_e \rightarrow \neg t)$, i.e., if the counter is $2^n - 1$ the tile is not t. The formula φ_2 is the disjunction of the following formulas.
   (a) The sequence does not start with tile s: $\neg s$.
   (b) Every 2^n multiple is not marked with tile t: $G\phi_t$.
   (c) The horizontal rule is violated somewhere that is not the end of a configuration and before the appearance of t at the end of a configuration:
   $$(\neg\phi_e \wedge (\bigwedge_{(p,p')\in H} \neg p \vee X\neg p'))R\phi_t$$
   (d) The propositions a and b are used to find a violation of the vertical rules. The disjunct of φ_2 that expresses a violation of the vertical rule is the conjunction of the following:
      i. The proposition a is assigned true exactly once, and b is assigned true after it: $F(a \wedge XG\neg a \wedge XFb)$.

ii. The proposition b is assigned *true* exactly once: $F(b \wedge XG\neg b)$.
iii. The location where b is assigned true occurs before a location where t marks a location that is a multiple of $2^n - 1$: $bR\phi_t$.
iv. The locations where a and b are true agree on the counter valuation: $G(a \rightarrow \bigwedge_{j=1}^{n} (q_j \leftrightarrow F(b \wedge q_j)))$.
v. There is exactly one occurrence of $2^n - 1$ between the locations where a and b are assigned true: $G(a \rightarrow \neg\phi_e U(\phi_e \wedge X(\neg\phi_e U b)))$
vi. The locations marked by a and b violate the vertical consistency:

$$\bigwedge_{(p,p')\in V} (F(a \wedge p) \rightarrow G(b \rightarrow \neg p'))$$

Lemma 2. *We have $M \models_{\preceq_L} \varphi$ iff T does not have a solution.*

Proof. Suppose that $M \models_{\preceq_L} \varphi$. It follows that for every possible sequence of tiles, we can find a truth assignment to the propositions in $\{a, b, q_1, \ldots, q_n\}$ such that this extension satisfies φ. It follows that the behavior of the propositions $\{q_1, \ldots, q_n\}$ is completely deterministic and simulates a 2^n-counter. Now, one of the disjuncts of φ_2 does not hold. It follows that there is some problem with the arrangement of tiles. Thus, all possible sequences of tiles are not valid and the tiling problem T does not have a solution.

In the other direction, suppose that the tiling problem T does not have a solution. Then, for every possible sequence or tiles there exists a problem with one of the tiling rules. We add the assignment of the propositions $\{a, b, q_1, \ldots, q_n\}$ according to this problem. □

In Appendix B we show also how to reduce the structure to one with a constant number of states. □

The next theorem states that $\preceq$ is a stronger relation than $\preceq_L$, which in turn helps explain why checking $\models_{\preceq}$ is more expensive than checking $\models_{\preceq_L}$.

Theorem 12. *For any partial Kripke structures M, M' and LTL formula φ, $M \preceq M'$ implies $M \preceq_L M'$, and therefore $M \models_{\preceq} \varphi$ implies $M \models_{\preceq_L} \varphi$.*

Proof. Immediate from the definitions of $\preceq$ and $\preceq_L$. □

Note that $\preceq$ is *strictly* stronger than $\preceq_L$, as the converse of the theorem does not hold. To illustrate this, consider the LTL formula $\varphi = (p \wedge Xp) \vee (\neg p \wedge X\neg p)$ and the partial Kripke structure M in Figure 1. Formally, let $M = \langle\{s_0, s_1, s_2\}, \{(s_0, s_1), (s_0, s_2), (s_1, s_1), (s_2, s_2)\}, L, s_0\rangle$ labeled with a single atomic proposition p such that $L(s_0, p) = \perp$, $L(s_1, p) = true$ and $L(s_2, p) = false$. It is easy to see that $[(M, s_0) \models \varphi] = \perp$. Moreover, we have $(M, s_0) \models_{\preceq_L} \varphi$, as every 3-valued trace generated from (M, s_0) can be completed by some 2-valued trace that satisfies φ. However, $(M, s_0) \not\models_{\preceq} \varphi$ as there does not exist a completion M' such that $M \preceq M'$ and $M' \models \varphi$, as state s_0 where $p = \perp$ cannot be completed to a *single* state s such that every trace from s satisfies φ: if $L(s, p) = true$, then the trace ss_2^ω violates φ, and if $L(s, p) = false$, then the trace ss_1^ω violates φ.

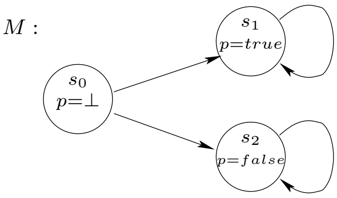

Fig. 1. Partial Kripke Structure showing that $\preceq$ is strictly stronger than $\preceq_L$.

5 Model Complexity

We have seen that LTL generalized model checking defined with the stronger branching-time preorder $\preceq$ is polynomial in the size of the model. The degree of the polynomial, however, is unbounded, and depends on the deterministic automaton created for the formula. Here we show that for interesting classes of properties, the model complexity can be restricted to linear or quadratic. The resemblance pointed out between generalized model checking and realizability in the proof of Theorem 8 continues here. Indeed, the same classes of formulas are used to suggest tractable fractions of LTL for realizability (cf. [RW89, AMPS98, PPS06]).

We start with a few additional definitions and known results regarding automata. Let $\mathcal{A} = \langle\Sigma, Q, q_{in}, \delta, \alpha\rangle$ be a Büchi automaton. We say that $\mathcal{A}$ is *weak* if there is a preorder $\leq$ on the state set Q such that the following two conditions hold:
1. For every $q \in Q$ and $\sigma \in \Sigma$, if q' appears in $\delta(q, \sigma)$ then $q \leq q'$.
2. For every $q \in Q$, if $q \in \alpha$ then forall q' such that $q \leq q'$ and $q' \leq q$ we have $q' \in \alpha$.

We use the acronyms mentioned previously for weak automata. For instance, an AWT is an alternating weak tree automaton and an DWW is a deterministic weak word automaton.
We specialize Theorem 4 to our needs as follows.

Theorem 13. *Given an APW A over a 1-letter alphabet, we can decide whether $L(A) = \emptyset$ in linear time if A is AWW [KVW00] and in quadratic time if A is an ABW, ACW, or an APW with three priorities [VW86, Jur00].*

Consider an LTL formula φ. We say that φ is a *safety property* if for every word $w \notin L(\varphi)$ there exists a prefix u such that forall v' we have $uv' \notin L(\varphi)$. Let p and q be Boolean combinations of propositional formulas. Formulas of the form GFp or $G(q \rightarrow Fp)$ are called *response properties*, and formulas of the form FGp are called *persistence properties* [MP92]. If φ is of the form $(\varphi_s^a \wedge \varphi_r^a) \rightarrow (\varphi_s^g \wedge \varphi_r^g)$ where φ_s^a and φ_s^g are conjunctions of safety properties and φ_r^a and φ_r^g are conjunctions of response properties is called *generalized reactivity[1]* [KPP03]. Alternatively, we classify LTL properties according to the type of deterministic automaton that accepts the same language. We say that φ is a *weak* *property* if there exists a DWW that accepts the language of φ. We say that φ is a *DBW property* if there exists a DBW that accepts the language of φ. Similarly, we say that φ is a *DCW property* if there exists a DCW that accepts the language of φ. The following theorem links the different types of LTL properties to the deterministic automata that accept them.

Theorem 14. *1. For every safety property φ, there exists a DWW D such that $L(D) = L(\varphi)$.*
2. For every response property φ, there exists a DBW D such that $L(D) = L(\varphi)$.
3. For every persistence property φ, there exists a DCW D such that $L(D) = L(\varphi)$.
4. For every generalized reactivity[1] property φ, there exists a DPW D with three priorities such that $L(D) = L(\varphi)$.

The following is a consequence of Theorems 13 and 14 and the proof of Theorem 6.

Theorem 15. *LTL generalized model checking $M \models_{\preceq} \varphi$ is linear in M for weak and safety properties, and quadratic in M for response, persistence, and generalized reactivity[1] properties.*

Proof. From the proof of Theorem 6 it follows that we combine a deterministic automaton for the property with the model to get an APW over a 1-letter alphabet. From Theorem 14 it follows that if the LTL property is a safety or obligation property the DPW, and the resulting APW, are weak. If the LTL property is a response property, the DPW is in fact a DBW. If the LTL property is a persistence property, the DPW is in fact a DCW. If the LTL property is a generalized reactivity[1] property, the DPW has three priorities. Recall that the APW is the product of the DPW and the model. Thus, the APW is linear in the size of the model. The desired upper bound now follows directly from Theorem 13. □

Note that LTL GMC for persistence properties can be solved in quadratic time in the size of the model, instead of in linear time as incorrectly stated in Theorem 5 of [GJ02]. The root cause of this error is the same as the one for Theorem 25 of [BG00], as the proofs of both theorems rely on the same product construction, now corrected in Theorem 6 of this paper.

Finally, we clarify a subtle misconception regarding generalized model checking of CTL properties. Given a CTL property, we can construct directly an NBT that is at most exponential in the size of the property that accepts all trees that satisfy the property [KVW00]. Generalized model checking can then be solved by combining this NBT with the model to obtain an ABW over a 1-letter alphabet [BG00]. According to Theorem 13 the emptiness of this ABW can be established in quadratic time. Thus, the complexity of GMC with respect to CTL properties is exponential in the formula and quadratic in the model, which is optimal [BG00]. As with LTL the quadratic complexity in the model follows from the type of acceptance condition used by the automaton for the formula. We are interested in classes of properties for which automata require simpler acceptance conditions. If the CTL property can be recognized by an NWT, the complexity in the size of the model reduces to linear. In the proof of Theorem 7 of [GJ02] it is assumed that if a CTL property can be recognized by an NCT then it can also be recognized by an NWT. However, it is currently unknown whether this is the case (cf. Section 6) and the proof of that theorem is therefore incomplete.

6 Conclusions

We study generalized model checking for linear time properties. We consider the classical definition of GMC and show that it is 2EXPTIME-complete in the size of the formula and polynomial in the structure. We study a linear version of the completeness preorder and show that this preorder induces a GMC problem that is EXPSPACE-complete in the size of the formula. We then proceed to show that for interesting classes of properties the model complexity can be restricted to a low order polynomial.

We have presented our work in the framework of partial Kripke structures. Other equally expressive 3-valued models [GJ03] include Modal Transition Systems [LT88] and Kripke Modal Transition Systems [HJS01]. All the complexity bounds given in this paper carry over to those closely related modeling formalisms.

The proof of Theorem 8 reduces realizability of LTL to GMC. The similarity actually goes in both directions. A GMC problem can be translated to a 2-person game where the specification (in LTL or in branching-time logic) can be translated to the winning condition. In a 2-person game players verifier and refuter alternate in moving a token along the edges of a graph. If the infinite path made by the token satisfies an LTL formula, verifier wins and otherwise she loses. If the winning condition is expressed in terms of branching-time logic, instead of considering a path in the graph, we consider the infinite unwinding of the game graph and prune the unwinding so that nodes that correspond to decisions of verifier have exactly one successor. The translation of the GMC problem to such a game is as follows. The game graph itself is similar to the model, where decisions of the refuter correspond to the branching of the original model and decisions of the verifier correspond to the values given to undetermined propositions. The formula to be checked on the model is translated to the winning condition in the game. Much like the proofs of the lower bounds above, this straightforward translation may result in a game graph that is exponential in the number of propositions whose value is unknown. In Appendix C we formally prove that LTL and CTL generalized model checking is linearly reducible to decision of a game G. The formal exposition is more complicated than explained above as we ensure that the size of the game graph is linear in the number of unknown propositions.

We have seen that for interesting classes of LTL and CTL properties the complexity in term of the model can be restricted to linear or quadratic. We classify the properties according to deterministic word automata and nondeterministic tree automata that match these formulas. While most popular types of properties are covered above, characterization of the exact classes of formulas that can be translated to these types of automata is an interesting problem. That is, what are the exact subsets of LTL that can be translated to DWW and to DBW? Is there a simple syntactic way to express these subsets? The same problem for CTL (and other branching-time logics) involves tree automata. For every CTL property there exist an NBT and an AWT recognizing the same set of trees [KVW00]. What CTL properties can be translated to NWT? Is there a syntactic way to express these subsets? We know that if a word language can be recognized by a DBW and by a DCW, then it can be recognized by a DWW [KMM04]. This suggests the following natural question: Given a tree language that is accepted by an NCT and by an NBT, can it be recognized by an NWT? From a practical point of view, it could be interesting to study the specific case of CTL properties that are recognized by NCT.

Acknowledgements. We thank Michael Huth for comments on an earlier version and Orna Kupferman for a discussion of the relative expressive power of NBT and NCT.

References

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *IFAC Symp. on System Structure and Control*, pages 469–474. Elsevier, 1998.

[BG99] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *11th Computer Aided Verification*, pages 274–287, 1999.

[BG00] G. Bruns and P. Godefroid. Generalized model checking: Reasoning about partial state spaces. In *11th Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 168–182, 2000.

[BR01] T. Ball and S. Rajamani. The SLAM Toolkit. In *13th Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 260–264, Paris, July 2001. Springer-Verlag.

[GC05] A. Gurfinkel and M. Chechik. How Thorough is Thorough Enough? In *13th Correct Hardware Design and Verification Methods*, 2005.

[GH05] P. Godefroid and M. Huth. Model Checking Vs. Generalized Model Checking: Semantic Minimizations for Temporal Logics. In *20th Logic in Computer Science*, pages 158–167, Chicago, June 2005.

[GHJ01] P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based Model Checking using Modal Transition Systems. In *12th Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 426–440, Aalborg, August 2001. Springer-Verlag.

[GJ02] P. Godefroid and R. Jagadeesan. Automatic Abstraction Using Generalized Model Checking. In *14th Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 137–150. Springer-Verlag, 2002.

[GJ03] P. Godefroid and R. Jagadeesan. On the Expressiveness of 3-Valued Models. In *4th Verification, Model Checking and Abstract Interpretation*, volume 2575 of *Lecture Notes in Computer Science*, pages 206–222, New York, January 2003. Springer-Verlag.

[GS97] S. Graf and H. Saidi. Construction of Abstract State Graphs with PVS. In *9th Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83, Haifa, June 1997. Springer-Verlag.

[GTW02] E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games*. Lecture Notes in Computer Science 2500. Springer-Verlag, 2002.

[GWC06] A. Gurfinkel, O. Wei, and M. Chechik. Systematic Construction of Abstractions for Model-Checking. In *7th Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 381–397. Springer-Verlag, January 2006.

[HJMS02] T. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy Abstraction. In *29th Principles of Programming Languages*, pages 58–70, 2002.

[HJS01] M. Huth, R. Jagadeesan, and D. Schmidt. Modal Transition Systems: a Foundation for Three-Valued Program Analysis. In *10th European Symp. on Programming*, volume 2028 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[Jur00] M. Jurdziński. Small progress measures for solving parity games. In *17th Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer-Verlag, 2000.

[Kle87] S. C. Kleene. *Introduction to Metamathematics*. North Holland, 1987.

[KMM04] O. Kupferman, G. Morgenstern, and A. Murano. Typeness for ω-regular automata. In *2nd Automated Technology for Verification and Analysis*, volume 3299 of *Lecture Notes in Computer Science*, pages 324–338. Springer-Verlag, 2004.

[KPP03] Y. Kesten, N. Piterman, and A. Pnueli. Bridging the gap between fair simulation and trace containment. In *15th Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 381–393. Springer-Verlag, 2003.

[KVW00] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[LT88] K. G. Larsen and B. Thomsen. A Modal Process Logic. In *3rd Logic in Computer Science*, pages 203–210, 1988.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, January 1992.

[Pit07] N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3):5, 2007.

[PPS06] N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive(1) designs. In *7th Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 364–380. Springer-Verlag, 2006.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *16th Principles of Programming Languages*, pages 179–190, 1989.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.

[Saf88] S. Safra. On the complexity of ω-automata. In *29th Foundations of Computer Science*, pages 319–327, White Plains, October 1988.

[SVW87] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[vEB97] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic and Recursion Theory*, volume 187 of *Lecture Notes in Pure and Applied Mathetaics*, pages 331–363, 1997.

[VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–221, April 1986.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

A Proof of Theorem 8

Theorem 8. *LTL Generalized model checking $M \models_{\preceq} \varphi$ is 2EXPTIME-hard in the size of φ.*

Proof. We show how to replace the exponential construction in Section 3 with a structure that has two states. The structure uses the propositions $\{p, z\} \cup O$. Let $n = |I|$. A state s in the structure above is represented by a sequence $s_1 \ldots s_n$ of n states. The assignment to i_j is the assignment to p in state s_j and the assignment to O is the assignment to O in state s_n. Specifically, $M = \langle S, R, L, s_p \rangle$ where $S = \{s_p, s_{\overline{p}}\}$, $R = S \times S$, and $L(s_p, p) = \mathit{true}$, $L(s_{\overline{p}}, p) = \mathit{false}$, and $L(s, q) = \bot$ for $s \in S$ and $q \in O \cup \{z\}$. The proposition z is going to mark the beginning of a sequence of n states.

We now rewrite the LTL formula φ into a new LTL formula that relativizes the occurrence of an input i_j in φ to the new location representing the value of i_j (by replacing i_j by $X^{j-1}p$), relativizes the occurrence of an output o in φ to the new location $X^{n-1}o$, and relativizes the truth values of all other operators to the locations where z is true. Formally, we define $g : LTL \to LTL$ as follows.
1. $g(i_j) = X^{j-1}p$ for $i_j \in I$
2. $g(o) = X^{n-1}o$ for $o \in O$
3. $g(\varphi \wedge \psi) = g(\varphi) \wedge g(\psi)$
4. $g(\neg\varphi) = \neg g(\varphi)$
5. $g(X\varphi) = X^n g(\varphi)$
6. $g(\varphi U \psi) = ((z \to g(\varphi)) U (z \wedge g(\psi)))$

Finally, let

$$\begin{aligned}\varphi_z = (\neg z \wedge Xz)\wedge \\ G(z \to (\underbrace{X(\neg z \wedge X(\neg z \wedge \ldots \wedge X(\neg z \wedge X}_{n \text{ next operators}}\, z))))))\end{aligned}$$

We set $\varphi' = \varphi_z \wedge Xg(\varphi)$. We note that the length of $g(\varphi)$ is linear in $|I| \cdot |\varphi|$ and the length of φ_z is linear in $|I|$.

We define a transformation on words that takes a word over $\mathbf{2}^{\{p,z\}\cup O}$ and produces a word over $\mathbf{2}^{I \cup O}$ that matches the transformation g for LTL formulas. Consider a word $w = \sigma_0\sigma_1 \cdots \in \mathbf{2}^{\{p,z\}\cup O}$. Let $\overline{g}(w)$ denote the word $w' = \sigma'_1 \cdots \in \mathbf{2}^{I\cup O}$ such that $\sigma'_m(i_j) = \sigma_{(m-1)n+j}(p)$ and $\sigma'_m(o) = \sigma_{mn}(o)$. Suppose that $w \models \varphi_z$. For every LTL formula φ and every location n we can show by induction on the structure of φ that $\overline{g}(w), n \models \varphi$ iff $w, (m-1)n + 1 \models g(\varphi)$. [3]

Suppose that φ is realizable by strategy f. We translate f to a labeling of the $\{0, 1\}^*$ tree in the obvious way. Every node $x0$ is labeled by $\neg p$ and every node $x1$ is labeled by p. Every node x such that $|x|$ is a multiple of n plus 1 is labeled by z and all others are labeled by $\neg z$. Every node x such that $|x|$ is a multiple of n is labeled by the appropriate assignment to the propositions in O. For every other node x, the assignment to propositions in O is arbitrary. Consider a path in the resulting tree. It is simple to see that the path satisfies φ_z. As the original path satisfies φ we can show that the result satisfies $Xg(\varphi)$ as well.

Suppose that there is a structure K that completes M such that $K \models \varphi'$. Consider the unwinding K^+ of K. As K is complete, so is K^+. Let $K^+ = \langle T, R', L', s^{in} \rangle$. We abuse notation and denote by K^+ a pruning of K^+ that includes for every state $t \in T$ exactly one state t' such that $(t, t') \in R'$ and $p \in L'(t')$ and exactly one state t'' such that $(t, t'') \in R'$ and $p \notin L'(t'')$. In addition, we identify a sequence $\pi \in \{0, 1\}^*$ with the state $t \in T$ such that ϵ is identified with s^{in} and $\pi 1$ is identified with the successor t' of π such that $p \in L'(t')$ and $\pi 0$ is identified with the successor t'' of π such that $p \notin L'(t'')$. Consider a letter $\sigma \in 2^I$, we identify σ with the sequence in $\alpha_1 \cdots \alpha_n \in \{0, 1\}^n$ such that $\alpha_j = 1$ iff $i_j \in \sigma$. Finally, we use the same notation for sequences in $(2^I)^*$. We are now ready to define the strategy $f : (2^I)^* \to 2^O$. For a sequence $\pi \in (2^I)^*$ we set $f(\pi) = L'_O(\pi)$. It is simple to see that every computation induced by f satisfies φ.

We note that it is possible also to reduce the number of propositions used in M to three by doing a similar encoding for the outputs using a single proposition q. The cycle induced by z has to grow to size $|I \cup O|$ and in addition the formula has to be modified to notice just the path labeled by p in the positions $(|I| + 1), \ldots, (|I| + |O|)$. □

B Proof of Theorem 11

Theorem 11. *LTL generalized model checking $M \models_{\preceq_L}$ with respect to the linear completeness preorder $\preceq_L$ is EXPSPACE-hard in the size of φ.*

Proof. In the construction in Section 4 the number of states of M is linear in the number of tiles of T and it uses n propositions. We show how to replace M with a system M' of a constant size that uses a constant number of propositions. The modifications to the LTL formula φ resemble those outlined above in the proof of Theorem 6. We are going to use the following propositions.
1. z – partial info – marks the location of the beginning of every tile.
2. p – full info – used to encode the name of the tile.
3. q – partial info – used to encodes the number of a tile module 2^n.

[3] Note that w starts from location 0 and $t(w)$ from location 1.

4. a, b – partial info – mark the locations where a vertical rule is violated (as above).

The system M' is the 2-state clique. One state is labeled p and the other $\neg p$. The state labeled p is initial. Formally, $M' = \langle S', R', L', s_p\rangle$, where $S' = \{s_p, s_{\overline{p}}\}$, $R' = S' \times S'$, and $L(s_p, p) = \mathit{true}$, $L(s_{\overline{p}}, p) = \mathit{false}$, and $L(s, r) = \perp$ for $s \in S$ and $r \in \{q, a, b, z\}$. Wlog, we assume that $n > i$. Then, a sequence of n states of this system encodes one tile and its number. Thus, a sequence of n states of M' corresponds to one state of M. The value of the proposition p in the states $i+1, \ldots, n$ is not used and thus there are many traces of M' that correspond to the same tiling. The case where $n \leq i$ is simple to construct using the same ideas.

We change the LTL formula φ into a new LTL formula that replaces the name of a tile with reference to a sequence of k truth values to proposition p, relativizes the value of the proposition q_i to the location representing the value of q_i (by replacing q_i by $X^{i-1}q$, and relativizes the truth values of all other operators to the locations where z is true. Formally, the function $g : LTL \to LTL$ is very similar to the function defined in the proof of Theorem 8 and is defined as follows.

1. Let $p_1, \ldots, p_k$ denote the binary encoding of the tile t. Let P_i be $X^{i-1}p$ if p_i is true and $X^{i-1}\neg p$ if p_i is false. Let $g(t) = \bigwedge_{i=1}^{k} P_i$ (note that $g(t)$ can be written with k next operators and not k^2 as above).
2. $g(q_i)$ be $X^{i-1}q$ for $q_i \in \{q_1, \ldots, q_n\}$
3. $g(r) = r$ for $r \in \{a, b\}$
4. $g(\varphi \wedge \psi) = g(\varphi) \wedge g(\psi)$
5. $g(\neg\varphi) = \neg g(\varphi)$
6. $g(X\varphi) = X^n g(\varphi)$
7. $g(\varphi U \psi) = ((z \to g(\varphi))U(z \wedge g(\psi)))$

Finally, let

$$\varphi_z = (\neg z \wedge Xz) \wedge \\ G(z \to (\underbrace{X\neg z \wedge X(\neg z \wedge \ldots \wedge X(\neg z \wedge X}_{n \text{ next operators}} z))))$$

We set $\varphi' = \varphi_z \wedge Xg(\varphi)$. We note that the length of φ' is linear in $n|\varphi|$.

The proof that $M' \models_{\preceq_L} \varphi'$ iff the tiling problem has no solution combines the proof of Lemma 2 with the techniques in the proof of Theorem 8. □

C From Generalized Model Checking to Two-player Games

We show how to use two-player games with temporal logic winning conditions to solve generalized model checking problems. We start with a definition of two player games and then show how to encode the GMC with respect to a given temporal logic formula as a game.

A *game* is $G = \langle V, V_1, V_2, E, L, v, \varphi\rangle$ where $\langle V, E\rangle$ is a graph, V_1 and V_2 form a partition of V to player 1 and player 2 vertices, respectively, $L : V \to 2^{AP}$ is a labeling of nodes with atomic propositions, $v \in V$ is an initial node, and φ is a winning condition for player 1 given in either LTL or CTL over propositions AP. A strategy σ for player 1 is $\sigma : V^* \cdot V_1 \to V$ such that for every $w \in V^*$ and $v_1 \in V_1$ we have $(v_1, \sigma(w \cdot v_1)) \in E$. A strategy τ for player 2 is defined similarly. Given a strategy σ for player 1, the outcome of σ from node v, denoted $out(\sigma, v)$, is the tree $T \subseteq V^*$ such that for every $w \in V^*$ and $v_1 \in V_1$ we have $\sigma(w \cdot v_1)$ is the unique son of $w \cdot v_1$ and for every $w \in V^*$ and $v_2 \in V_2$ the nodes $\{w \cdot v_2 \cdot v' \mid (v_2, v') \in E\}$ is the set of sons of $w \cdot v_2$. Given a strategy σ for player 1, a strategy τ for player 2, and a node v, the outcome of σ and τ from v, denoted $out(\sigma, \tau, v)$, is the word $w = v_0 v_1 \cdots \in V^\omega$ such that for every $i \geq 0$ we have if $v_i \in V_1$ then $v_{i+1} = \sigma(v_0 \cdots v_i)$ and if $v_i \in V_2$ then $v_{i+1} = \tau(v_0 \cdots v_i)$. Given a winning condition φ, a strategy σ is winning for player 1 from node v if φ is in CTL and $out(\sigma, v) \models \varphi$ and if φ is in LTL and for every strategy τ for player 2 we have $out(\sigma, \tau, v) \models \varphi$. Player 1 wins G if she has a winning strategy from the initial node v. We decide a game G by determining whether player 1 can win the game.

Theorem 16. *LTL and CTL generalized model checking $M \models_{\preceq} \varphi$ is linearly reducible to decision of a game G.*

Proof. Consider the PKS $M = \langle S, R, L, s^{in}\rangle$ and the formula φ. Let $AP = \{p_1, \ldots, p_n\}$. Define the game $G = \langle V, V_1, V_2, E, L', v^{in}, \varphi'\rangle$ as follows. Let $V' = S \times [n] \times \{\mathit{true}, \mathit{false}\}$. Given a node $v = (s, i, \alpha) \in V'$ we say that v is *consistent* if either $\alpha = L(s, p_i)$ or $L(s, p_i) = \perp$. The set of nodes V is $S \cup \{v \in V' \mid v \text{ is consistent}\}$ and $V_1 = S \cup (V \cap (S \times [n-1] \times \{\mathit{true}, \mathit{false}\}))$ and $V_2 = V \cap (S \times \{n\} \times \{\mathit{true}, \mathit{false}\})$. The set of edges E is defined as follows. For a node $s \in V_1$ we have $(s, (s, 1, \beta)) \in E$ forall β such that $(s, 1, \beta) \in V$. For a node $v = (s, i, \alpha) \in V_1$ we have $((s, i, \alpha), (s, i+1, \beta)) \in E$ forall β such that $(s, i+1, \beta) \in V$. For a node $v = (s, n, \alpha) \in V_2$ we have $((s, n, \alpha), s')$ forall s' such that $(s, s') \in R$. Finally, L' is defined over $AP' = AP \cup \{q\}$ where L' is defined as follows.

$$L'(v, r) = \begin{cases} \mathit{true} \text{ If } r = p_i \text{ and } v = (s, i, \mathit{true}) \\ \mathit{false} \text{ If } r = p_i \text{ and } v = (s, i, \mathit{false}) \\ \mathit{false} \text{ If } r \neq p_i \text{ and } v = (s, i, \alpha) \\ \mathit{true} \text{ If } r = q \text{ and } v \in S \end{cases}$$

Suppose that φ is an LTL formula. Then, φ' is obtained from φ by replacing (recursively) p_i by $X^i p_i$, replacing $X\psi$ by $q \wedge X(\neg q U(q \wedge \psi))$, and $\psi_1 U \psi_2$ by $(q \to \psi_1)U(q \wedge \psi_2)$.

Suppose that φ is a CTL formula. Then φ' is obtained from φ by replacing (recursively) p_i by $EX^i p_i$, replacing $EX\psi$ by $q \wedge EXE(\neg q U(q \wedge \psi))$, replacing $AX\psi$ by $q \wedge AXA(\neg q U(q \wedge \psi))$, replacing $E(\psi_1 U \psi_2)$ by $E((q \to \psi_1)U(q \wedge \psi_2))$, and replacing $U(\psi_1 U \psi_2)$ by $A((q \to \psi_1)U(q \wedge \psi_2))$.

In both cases it is possible to show that a winning strategy σ for player 1 translates to a completion M' of M such that M' satisfies φ and that a completion M' of M translates to a winning strategy for player 1. It follows that player 1 wins iff GMC of M with respect to φ holds.

The number of nodes of G can be further reduced to twice the number of states of M by encoding all the propositions in

AP by one proposition p in G and augmenting φ by clauses that force player 1 to leave cycles where q does not hold.