Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 91 (2004) 201–210
Information Processing Letters
www.elsevier.com/locate/ipl

The power of first-order quantification over states in branching and linear time temporal logics

Krishnendu Chatterjee [a], Pallab Dasgupta [b,*], P.P. Chakrabarti [b]

[a] *Department of EECS, UC Berkeley, USA*
[b] *Department of Computer Science & Engineering, Indian Institute of Technology, Kharagpur 721302, India*

Received 1 October 2003; received in revised form 5 May 2004

Available online 17 June 2004

Communicated by K. Iwama

Abstract

In this paper, we investigate the power of extending first-order quantification over states to branching and linear time temporal logics. We show that an unrestricted extension significantly enriches the expressive power of μ-calculus, but also leads to a significant jump in model checking complexity. However, by restricting the scope of the extension, we are able to present a powerful extension of μ-calculus that is richer than μ-calculus, but is in the same complexity class as μ-calculus in terms of model checking complexity. In the case of linear time temporal logic, we find that first-order quantification over states is more computationally expensive. We show that even under the most restricted scope of quantification, the program complexity of model checking linear temporal logic (LTL) is NP-hard and coNP-hard. However, we also show that model checking LTL with this generic extension remains PSPACE-complete.

Keywords: Model checking; Verification; Linear temporal logic; Formal methods

1. Introduction

Model checking [1] has emerged as one of the most powerful techniques for formal property verification in hardware and other reactive systems. The increasing acceptance of formal property verification within the validation flow of protocol and chip design has catalyzed the formalization of several languages for formal property specification, such as Forspec (of Intel), Sugar (of IBM/Accelera) and OVA (of Synopsys). These languages have been built around the syntactic structure of known branching time and linear time temporal logics such as *Computation Tree Logic* (CTL) and *Linear Temporal Logic* (LTL) [1]. In recent times, the different fora at which these language standards are being crystallized have felt the necessity to investigate whether the power of these known temporal logics can be further enriched at low overhead in terms of model checking complexity.

* Corresponding author.
E-mail address: pallab@cse.iitkgp.ernet.in (P. Dasgupta).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.05.003

In this paper, we investigate the power of extending first-order quantification over states to the generic branching time and linear time temporal logics. The research is motivated by our initial results on this subject [7], which show that a well constrained use of first-order quantification over states in CTL vastly enriches its expressive power without any significant blow-up in model checking complexity. This result opens up interesting possibilities, since other forms of first-order quantification in temporal logics (such as [6,9]) exhibit a blow-up in model checking complexity.

While our early results are promising, they do not indicate the power of quantification over branching time properties in general, nor do they provide any insights to the effect of similar extensions to linear time temporal logics. This paper provides a series of results which attempts to answer most of these questions, and shows how we may effectively utilize the power of first-order quantification over states at low computational overhead.

Specifically, we investigate the effect of extending first-order quantification over states to μ-calculus [3,5] (branching-time) and LTL (linear time). Since most branching time temporal logics are subsumed by propositional μ-calculus [2], investigating the effect of first-order quantification over states on μ-calculus gives us a good indication of its power over branching time temporal logics in general. The model-checking problem for μ-calculus is in NP $\cap$ coNP, but there are several fragments of μ-calculus for which efficient model checking algorithms exist [4].

The notion of first-order quantification over states may be demonstrated by the following example. Suppose we wish to determine whether there exists some state t such that (a) the proposition q is true at t, (b) there exists a path from a state s to t through states labeled by proposition p, and (c) there exists a path from s to t through states labeled by proposition r. This property cannot be expressed in propositional μ-calculus. The μ-calculus formula: $\mu y.(q \vee (p \wedge EX(y))) \wedge \mu y.(q \vee (r \wedge EX(y)))$ does not express the same intent, as it does not require us to find paths to a common state t where q is true. By extending first-order quantification over states, we can express the desired intent as:

$$\exists x \in q.\big[\mu y.\big(x \vee \big(p \wedge EX(y)\big)\big) \wedge \mu y.\big(x \vee \big(r \wedge EX(y)\big)\big)\big].$$

The requirement of finding a common state t in the two qualified paths is enforced through the binding of the common variable x in the two subformulas. We present several other examples in the paper highlighting the added expressibility due to first-order quantification over states (showing both existential and universal quantification). The contributions of this paper are as follows:

(1) We show that an unrestricted use of first-order quantification over states significantly enriches the expressive power of μ-calculus, but also leads to a significant jump in model checking complexity (PSPACE-complete).
(2) By restricting the scope of the extension, we are able to present a powerful extension of μ-calculus (called Scope Restricted Quantified μ Calculus) that is richer than μ-calculus, but is in the same complexity class (NP $\cap$ coNP) as μ-calculus in terms of model checking complexity. We also show that if μ-calculus model checking is in P then Scope Restricted Quantified μ Calculus model checking is also in P.
(3) In the case of linear time temporal logic, we find that first-order quantification over states is more computationally expensive. We show that even under the most restricted scope of quantification, the program complexity of model checking linear temporal logic (LTL) is NP-hard and coNP-hard.
(4) We also show that model checking the generic extension of LTL with first-order quantification over states remains PSPACE-complete.

2. Examples of first-order quantification in μ-calculus

This section demonstrates the expressibility of the proposed extension, Scope Restricted Quantified μ Calculus through two examples. We hope that these examples will informally explain the intent of the proposed semantics before we present the logic in formal terms.

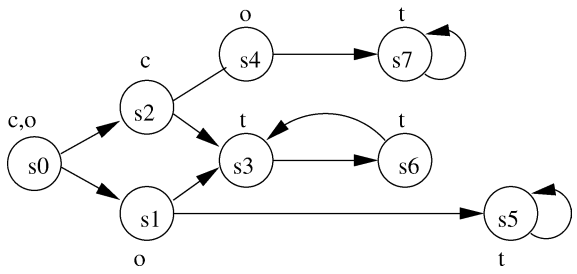

Fig. 1. A sample state transition system.

Example 2.1 (*Few interesting properties not expressible in μ-calculus*). Consider the labeled transition system shown in Fig. 1. States labeled by atomic propositions t, c and o respectively denote *terminal*, *control* and *observable* states. The following illustrative properties are expressible in Scope Restricted Quantified μ Calculus but not in μ-calculus.

- Are all terminal states reachable? This is expressed as $\psi_1 = \forall x \in t.[\mu y.(x \vee EX(y))]$. The property is true at s_0 but false at s_2 since the terminal state s_5 is not reachable from s_2.
- For every terminal state t' is there a path such that t' occurs infinitely often (Büchi property) along the path? The formula, $\psi_2 = \forall x \in t.[\nu y_1 \mu y_2.(EX(y_1) \wedge (EX(y_2) \vee x))]$, expresses the desired property. The property is true at s_0 but false at s_2.
- Is there a terminal state t' such that there exists a path along which eventually t' occurs forever (coBüchi property)? This formula, $\psi_3 = \exists x \in t.[\mu y_1 \nu y_2.(EX(y_1) \vee ((EX(y_2) \wedge x))]$, expresses the desired property. The property is true at s_0 and s_2. The path $s_0 s_1 (s_5)^\omega$ is a witness for the truth of ψ_3 at s_0, while the path $s_2 s_4 (s_7)^\omega$ is a witness for its truth at s_2.
- Is there a state satisfying ψ_1 that is reachable along a controllable path and also an observable path? The formula, $\psi_4 = \exists x \in \psi_1.[\mu y.(x \vee (c \wedge EX(y))) \wedge \mu y.(x \vee (o \wedge EX(y)))]$, expresses the desired property.

Example 2.2 (*Gaming properties*). The state space of the game may be modeled as game graphs of the form $G = (V, E)$. The vertex set V is partitioned into two sets V_p and V_ad. At the vertices $v \in V_\mathrm{p}$ the player chooses the successor, while the choice of the successor is with the adversary at the vertices $v \in V_\mathrm{ad}$. Some of the vertices are defined to be *goal vertices*. We model the game graph as a Kripke structure M with $\mathcal{S} = V, \mathcal{R} = E$. The atomic propositions are v_p, v_ad and g. Each vertex $v \in V_\mathrm{p}$ is labeled by v_p, each vertex in V_ad is labeled by v_ad and the vertices which are the *goal vertices* are labeled by g. A strategy for the player (respectively adversary) is a choice of successors from the vertices in the set V_p (respectively V_ad). We present some interesting properties expressible in Scope Restricted Quantified μ Calculus on such game graphs:

- For every goal vertex s' is there a player strategy such that for any strategy of the adversary s' occurs infinitely often? To express the property, we first define $\langle\langle p\rangle\rangle X(y) = (v_\mathrm{p} \wedge EX(y)) \vee (v_\mathrm{ad} \wedge AX(y))$. Intuitively, given a set of vertices denoted by the variable y, $\langle\langle p\rangle\rangle X(y)$ denotes the set of states from which the player is able to reach the set y in one step regardless of the move of the adversary. The desired intent is expressed in Scope Restricted Quantified μ Calculus by the formula $\psi_5 = \forall x \in g.[\nu y_1 \mu y_2.(\langle\langle p\rangle\rangle X(y_1) \wedge (\langle\langle p\rangle\rangle X(y_2) \vee x))]$.
- Is there a specific goal vertex s' such that the player can reach s' regardless of the moves of the opponent? The formula to express the property is $\psi_6 = \exists x \in g.[\mu y_1.(x \vee \langle\langle p\rangle\rangle X(y_1))]$.
- For every goal vertex s', does there exist a strategy to reach s' regardless of the opponent? The formula, $\psi_7 = \forall x \in g.[\mu y_1.(x \vee \langle\langle p\rangle\rangle X(y_1))]$, expresses the desired property.

3. Scope restricted quantified μ-calculus

The semantics of the proposed logic is interpreted over a Kripke structure defined as follows:

Definition 3.1 (*Kripke structure*). A Kripke structure is a tuple $M = \langle \mathcal{AP}, \mathcal{S}, \mathcal{R}, s_0, \mathcal{F} \rangle$ where $\mathcal{AP}$ is the set of atomic propositions, $\mathcal{S}$ is the finite set of states, $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation such that $(s_i, s_j) \in \mathcal{R}$ if s_j is a successor of s_i, $s_0 \in \mathcal{S}$ is the initial state, $\mathcal{F} : \mathcal{S} \to 2^{\mathcal{AP}}$ is a function labeling states with atomic propositions true in them.

A Quantified μ Calculus formula may have two types of variables, namely first-order variables that are quantified by the $\exists$ or $\forall$ quantifiers, and (second order) μ-calculus variables whose fixpoints are computed using μ or ν. A generic Quantified μ Calculus formula may have a subformula of the form $\phi(Z, \mathcal{L})$, where Z denotes the set of free μ-calculus variables (quantified by μ / ν from the outside), and $\mathcal{L}$ denotes the set of free first-order variables. In the scope restricted version we prevent the scopes of first order variables to overlap, therefore $\mathcal{L}$ has at most one member for each subformula. This choice is natural, since alternation between first-order variables is a well established source of complexity. We also disallow the first-order quantification of a subformula that has free μ-calculus variables, otherwise the semantics of the fixpoint computation becomes too complex for meaningful usage in practice. The syntax of the logic is as follows. ϕ (which is equivalent to $\phi([\,], \_)$) represents the set of Scope Restricted Quantified μ Calculus formulas (in negation normal form).

$$
\begin{aligned}
&\phi ::= p \text{ where } p \in \mathcal{AP} \mid \neg p \mid \phi \; op \; \phi \mid EX[\phi] \mid AX[\phi] \\
&\qquad \mid \nu y.\phi([y], [\,]) \mid \mu y.\phi([y], [\,]) \mid \exists x \in \phi.[\phi([\,], x)] \mid \forall x \in \phi.[\phi([\,], x)], \\
&\phi(Z, [\,]) ::= \phi, \text{ if } Z = [\,] \mid y, \text{ if } Z = [y] \mid \phi(Z_1, [\,]) \; op \; \phi(Z_2, [\,]) \text{ where } Z_1, Z_2 \subseteq Z \\
&\qquad \mid EX[\phi(Z, [\,])] \mid AX[\phi(Z, [\,])] \mid \nu y.\phi([y||Z], [\,]) \mid \mu y.\phi([y||Z], [\,]), \\
&\phi(Z, x) ::= x, \text{ if } Z = [\,] \mid \phi(Z_1, x) \; op \; \phi(Z_2, x) \text{ where } Z_1, Z_2 \subseteq Z \\
&\qquad \mid EX[\phi(Z, x)] \mid AX[\phi(Z, x)] \mid \nu y.\phi([y||Z], x) \mid \mu y.\phi([y||Z], x), \\
&op ::= \vee \mid \wedge.
\end{aligned}
$$

The $||$ operator is used to denote the concatenation of lists. $[\,]$ denotes the empty list. The main restrictions to the generalized use of quantification over states are summarized as follows:

(1) We do not allow first-order quantification over a subformula having a free first-order variable (to contain the complexity).
(2) We do not allow first-order quantification over a subformula having a free μ-calculus variable (to simplify the semantics).
(3) The first-order variables are instantiated by formulas that have no free variables. In formulas of the form $\exists x \in \phi \ldots$ and $\forall x \in \phi \ldots$ the subformula ϕ does not have a free variable.

It may be noted that these restrictions do not completely prevent us from using nesting of quantifiers (as long as there is no overlap between scopes of free variables). For example the formula, ψ_4 in Example 2.1, exhibits the nesting of quantifiers. Also in the formula:

$$\exists x \in \psi_1.\big[\mu y.\big(x \vee \big(\psi_2 \wedge EX[y]\big)\big)\big]$$

the subformulas ψ_1 and ψ_2 can have other quantifiers, since they are free of the first-order variable x and the μ-calculus variable y. On the other hand, the formula:

$$\exists x_1 \in t.\forall x_2 \in c.\big[\mu y.\big(x_1 \vee \big(x_2 \wedge EX[y]\big)\big)\big]$$

is not a valid Scope Restricted Quantified μ Calculus formula.

A formula ϕ with free variables $y_1, y_2, \ldots, y_n$ is interpreted as a mapping from $(2^{\mathcal{S}})^n \to 2^{\mathcal{S}}$. We write $\phi(y_1, y_2, \ldots, y_n)$ to denote that all the free variables of ϕ are among $y_1, y_2, \ldots, y_n$. A valuation $\mathcal{E} = (V_1, V_2, \ldots, V_n)$ is an assignment of the subsets of $\mathcal{S}$, $V_1, V_2, \ldots, V_n$ to free propositional variables $y_1, y_2, \ldots, y_n$ respectively.

$[\![\phi]\!]_{\mathcal{E}}$ denotes the evaluation of ϕ on the actual arguments $V_1, V_2, \ldots, V_n$. The free state variable x can be instantiated to a state s, in which case we label s with x treating x as an atomic proposition true only at s. Hence instantiation $\mathcal{I}$ is an labeling function $\mathcal{I}: x \to \mathcal{S}$ which labels x to exactly one state. $[\![\phi'(x)]\!]_{\mathcal{E}\mathcal{I}}$ denotes the evaluation of $\phi'(x)$ under the valuation $\mathcal{E}$ and instantiation $\mathcal{I}$.

The semantics of Scope Restricted Quantified μ Calculus is as follows:

- $[\![p]\!]_{\mathcal{E}} = \{s\colon s \in \mathcal{S} \text{ and } p \in \mathcal{F}(s)\}$ where $p \in \mathcal{AP}$,
- $[\![y_i]\!]_{\mathcal{E}} = V_i$,
- $[\![EX(\phi(Z, [\,]))]\!]_{\mathcal{E}} = \{s\colon \exists t \in [\![\phi(Z, [\,])]\!]_{\mathcal{E}} \text{ and } (s,t) \in \mathcal{R}\}$,
- $[\![\phi_1 \wedge \phi_2]\!]_{\mathcal{E}} = [\![\phi_1]\!]_{\mathcal{E}} \cap [\![\phi_2]\!]_{\mathcal{E}}$,
- $[\![AX(\phi(, Z, [\,]))]\!]_{\mathcal{E}} = \{s\colon \forall t (s,t) \in \mathcal{R} \to t \in [\![\phi(Z, [\,])]\!]_{\mathcal{E}}\}$,
- $[\![\mu y_i.\phi(y_i||Z, [\,])]\!]_{\mathcal{E}} = \bigcap\{S' \subseteq \mathcal{S}\colon [\![\phi(y_i||Z, [\,])]\!]_{\mathcal{E}[y_i \leftarrow S']} \supseteq S'\}$,
- $[\![\nu y_i.\phi(y_i||Z, [\,])]\!]_{\mathcal{E}} = \bigcup\{S' \subseteq \mathcal{S}\colon [\![\phi(y_i||Z, [\,])]\!]_{\mathcal{E}[y_i \leftarrow S']} \supseteq S'\}$,
- $[\![\exists x \in \phi.[\phi'([\,], x)]]\!]_{\mathcal{E}} = \bigcup\{[\![\phi'([\,], x)]\!]_{\mathcal{E}\mathcal{I}}\colon \mathcal{I}(x) \in [\![\phi]\!]_{\mathcal{E}}\}$,
- $[\![x]\!]_{\mathcal{E}\mathcal{I}} = \{\mathcal{I}(x)\}$,
- $[\![\forall x \in \phi.[\phi'([\,], x)]]\!]_{\mathcal{E}} = \bigcap\{[\![\phi'([\,], x)]\!]_{\mathcal{E}\mathcal{I}}\colon \mathcal{I}(x) \in [\![\phi]\!]_{\mathcal{E}}\}$,
- $[\![\phi'(Z, x) \wedge \phi]\!]_{\mathcal{E}\mathcal{I}} = [\![\phi'(Z, x)]\!]_{\mathcal{E}\mathcal{I}} \cap [\![\phi]\!]_{\mathcal{E}}$,
- $[\![\phi'_1(Z_1, x) \wedge \phi'_2(Z_2, x)]\!]_{\mathcal{E}\mathcal{I}} = [\![\phi'_1(Z_1, x)]\!]_{\mathcal{E}\mathcal{I}} \cap [\![\phi'_2(Z_2, x)]\!]_{\mathcal{E}\mathcal{I}}$,
- $[\![EX(\phi'(Z, x))]\!]_{\mathcal{E}\mathcal{I}} = \{s\colon \exists t \in [\![\phi'(Z, x)]\!]_{\mathcal{E}\mathcal{I}} \text{ and } (s,t) \in \mathcal{R}\}$,
- $[\![AX(\phi'(Z, x))]\!]_{\mathcal{E}\mathcal{I}} = \{s\colon \forall t (s,t) \in \mathcal{R} \to t \in [\![\phi'(Z, x)]\!]_{\mathcal{E}\mathcal{I}}\}$,
- $[\![\mu y_i.\phi'(y_i||Z, x)]\!]_{\mathcal{E}\mathcal{I}} = \bigcap\{S' \subseteq \mathcal{S}\colon [\![\phi'(y_i||Z, x)]\!]_{\mathcal{E}[y_i \leftarrow S']\mathcal{I}} \supseteq S'\}$,
- $[\![\nu y_i.\phi'(y_i||Z, x)]\!]_{\mathcal{E}\mathcal{I}} = \bigcup\{S' \subseteq \mathcal{S}\colon [\![\phi'(y_i||Z, x)]\!]_{\mathcal{E}[y_i \leftarrow S']\mathcal{I}} \supseteq S'\}$.

The semantic rules for the $\vee$ operator is similar to that of $\wedge$ with $\cup$ replacing $\cap$ in the RHS. We now show that the model checking problem of Scope Restricted Quantified μ Calculus lies in the same complexity class as that of μ-calculus model checking. In the following proofs we denote $|M| = |\mathcal{S}| + |\mathcal{R}|$ as the size of the Kripke structure.

Lemma 3.1. *If μ-calculus model checking is in* P *then so is Scope Restricted Quantified μ Calculus model checking.*

Proof. Suppose there is a polytime algorithm for μ-calculus model checking that has a complexity of $O((|M|.|f|)^k)$ where $|f|$ is the length of the μ-calculus formula and k is some constant (without loss of generality $k \geqslant 1$). We prove that Scope Restricted Quantified μ Calculus model checking can work in $O(|\mathcal{S}|.(|M|.|f|)^k)$ time.

Given an instantiation, $\mathcal{I}$, of the state variable x, the problem of evaluating $\phi'(x)$ is equivalent to evaluating the μ-calculus formula obtained by treating x as an atomic proposition in $\phi'(x)$ which is true only at $\mathcal{I}(x)$. Using the given algorithm for μ-calculus model checking, we can evaluate $\phi'(x)$ in $O((|M|.|f|)^k)$ time. We now use induction on the length of the formula to complete the proof. Consider a formula $\psi = \exists x \in \varphi.[\varphi'(x)]$. Since each subformula of φ is independent of x we can evaluate φ in $O(|\mathcal{S}|.(|M|.|f_1|)^k)$ time, where $|f_1|$ is the length of φ. Once this labeling is done for all the subformulas of ϕ, we treat these as atomic propositions labeling the states of the system. The remaining formulas are of the form $\phi'(x)$. Thus once the labeling of the formulas of the form ϕ is done, for each instantiation of x we can evaluate $\varphi'(x)$ in $O((|M|.|f_2|)^k)$ time, where $|f_2|$ is the length of $\varphi'(x)$. Since there can be at most $|\mathcal{S}|$ instantiations of x we can evaluate ψ in $O(|\mathcal{S}|.(|M|.|f_1|)^k) + O(|\mathcal{S}|.(|M|.|f_2|)^k)$ time (also as $k \geqslant 1$ we have $|f_1|^k + |f_2|^k \leqslant (|f_1| + |f_2|)^k$). Hence we have $O(|\mathcal{S}|.(|M|.|f_1|)^k) + O(|\mathcal{S}|.(|M|.|f_2|)^k) \leqslant O(|\mathcal{S}|.(|M|.|f|)^k)$. A similar argument applies when the quantifier is $\forall$ instead of $\exists$. □

Given a μ-calculus formula ϕ, the alternation depth of a propositional variable y refers to the "significant" nesting of alternating μ's and ν's binding y. Alternation depth of ϕ is the maximum alternation depth among its propositional variables [3].

Lemma 3.2. *Scope Restricted Quantified μ Calculus with alternation depth bounded by l can be checked in* $O(|\mathcal{S}|.(|M|.|f|)^{l+1})$ *time.*

Proof. There is a known algorithm to check μ-calculus formulas with alternation depth bounded by l in $O((|M|.|f|)^{l+1})$ time [3]. Using an argument similar to that in Lemma 3.1, it is easy to prove that Scope Restricted Quantified μ Calculus formulas with alternation depth bounded by l can be checked in $O(|\mathcal{S}|.(|M|.|f|)^{l+1})$ time. □

Theorem 3.1. *Scope Restricted Quantified μ Calculus model checking is in* NP ∩ co NP.

Proof. We first consider properties with the ∃ quantifier. The argument for properties with the ∀ quantifier is similar. Let M be a Kripke structure with $|\mathcal{S}|$ as the size of the state space and $|\mathcal{R}|$ as the size of the transition relation. Given M and a formula of the form $\psi = \exists x \in \phi.[\mu y.g(y,x)]$ we guess annotations of M with the subformulas of ψ true at each state corresponding to every possible instantiation of x. There can be at most $|\mathcal{S}|$ instantiations as ϕ can be true in at most $|\mathcal{S}|$ states. Each annotation for a given instantiation provides the "rank" for each μ variable y indicating how many times the associated μ formula $\mu y.g(y)$ is unwound for the given instantiation. (These ranks corresponds to the indices in the Tarski–Knaster sequence of approximations for the least fixed point.) Thus given an instantiation of x we might have ranked μ variable y^5, which is equivalent to $g(y^4)$ at a state s depending on y^4, equivalent to $g(y^3)$ at a state t depending on y^3 at a state u and so forth; the depending on relation should be well founded as $\mu y.g(y)$ is the least fixed point and can be unwound bounded (viz. $|\mathcal{S}|$) number of times. In general for every instantiation the ranks will be tuples of natural numbers of value at most $|\mathcal{S}|$ and there can be at most $|\mathcal{S}|$ instantiations. The general idea is explained in [3], but here we need the annotation to be guessed for every instantiation of the state variable x. Intuitively $\exists x \in \phi.[\mu.g(y,x)]$ is equivalent to $\bigvee_{s_i \in [\![\phi]\!]} \mu.g(y,x_i)$ where $g(y,x_i)$ is the formula with x_i treated as an atomic proposition true only in state s_i. The ranks are ordered lexicographically. After guessing ranked, threaded annotation for every instantiation we simply verify that it is well founded. This shows that Scope Restricted Quantified μ Calculus model checking is in NP. Membership in co NP follows from the fact that Scope Restricted Quantified μ Calculus is closed under complementation. □

4. Generalized quantified μ-calculus

We now study the effect of allowing an unrestricted use of first-order quantification. We call the following extension *Generalized Quantified μ Calculus*. $\mathcal{L}$ denotes a list of state variables.

$$\begin{aligned}
\phi(Z,\mathcal{L}) ::= {} & \phi(Z,\mathcal{L}) \vee \phi(Z,\mathcal{L}) \mid \phi(Z,\mathcal{L}) \wedge \phi(Z,\mathcal{L}) \mid EX(\phi(Z,\mathcal{L})) \mid AX(\phi(Z,\mathcal{L})) \mid \\
& \mu y.\phi(y||Z,\mathcal{L}) \mid \nu y.\phi(y||Z,\mathcal{L}) \mid p \mid \neg p \mid y \quad \text{where } p \in \mathcal{AP} \text{ and } y \in \mathcal{Y}, \\
Q(Z,\mathcal{L}) ::= {} & \exists x \in Q(Z,\mathcal{L}).[Q(Z,x||\mathcal{L})] \mid \forall x \in Q(Z,\mathcal{L}).[Q(Z,x||\mathcal{L})] \text{ where } x \notin \mathcal{L} \\
& \mid Q(Z,\mathcal{L}) \wedge Q(Z,\mathcal{L}) \mid Q(Z,\mathcal{L}) \vee Q(Z,\mathcal{L}) \mid EX(Q(Z,\mathcal{L})) \mid AX(Q(Z,\mathcal{L})) \mid \phi(Z,\mathcal{L}) \\
& \mid \mu y.Q(y||Z,\mathcal{L}) \mid \nu y.Q(y||Z,\mathcal{L}) \mid x \quad \text{where } x \in \mathcal{X}.
\end{aligned}$$

Generalized Quantified μ-calculus formulas are of the form $Q([\,],[\,])$, where $[\,]$ stands for the empty list. The evaluation of $Q(Z,\mathcal{L})$ is subject to an instantiation of the free state variables in its list $\mathcal{L}$ and an environment for the variables in Z. A free state variable $x \in \mathcal{L}$ can be instantiated to a state s in which case we label s with x and treat x as an atomic proposition true only at s. Thus an instantiation of $\mathcal{L}$ can be viewed as a labeling function $\mathcal{I}: \mathcal{L} \to \mathcal{S}$, where each label $x \in \mathcal{L}$ is used exactly in one state. For subformulas without free state variables $\mathcal{I}$ is always empty.

The semantics of Generalized Quantified μ-calculus properties are defined in terms of a valuation $\mathcal{E}$, and an instantiation $\mathcal{I}$. $\mathcal{I}_1$ and $\mathcal{I}_2$ are $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively.

- $[\![p]\!]_{\mathcal{EI}} = \{s\colon\ s \in \mathcal{S} \text{ and } p \in \mathcal{F}(s)\}$,
- $[\![y_i]\!]_{\mathcal{EI}} = \mathcal{E}(y_i)$,
- $[\![x_i]\!]_{\mathcal{EI}} = \mathcal{I}(x_i)$,
- $[\![\psi_1(Z_1, \mathcal{L}_1) \wedge \psi_2(Z_2, \mathcal{L}_2)]\!]_{\mathcal{EI}} = [\![\psi_1(Z_1, \mathcal{L}_1)]\!]_{\mathcal{EI}_1} \cap [\![\psi_2(Z_2, \mathcal{L}_2)]\!]_{\mathcal{EI}_2}$,
- $[\![EX(\psi(Z, \mathcal{L}))]\!]_{\mathcal{EI}} = \{s\colon\ \exists s' \text{ such that } (s, s') \in \mathcal{R} \text{ and } s' \in [\![(\psi(Z, \mathcal{L})]\!]_{\mathcal{EI}}\}$,
- $[\![AX(\psi(Z, \mathcal{L})]\!]_{\mathcal{EI}} = \{s\colon\ \forall s' \text{ if } (s, s') \in \mathcal{R} \text{ then } s' \in [\![(\psi(Z, \mathcal{L})]\!]_{\mathcal{EI}}\}$,
- $[\![\mu y.\phi'(y||Z, \mathcal{L})]\!]_{\mathcal{EI}} = \bigcap\{S' \subseteq \mathcal{S}\colon\ [\![\phi'(y||Z, \mathcal{L})]\!]_{\mathcal{E}[y \leftarrow S']\mathcal{I}} \supseteq S'\}$,
- $[\![\nu y.\phi'(y||Z, \mathcal{L})]\!]_{\mathcal{EI}} = \bigcup\{S' \subseteq \mathcal{S}\colon\ [\![\phi'(y||Z, \mathcal{L})]\!]_{\mathcal{E}[y \leftarrow S']\mathcal{I}} \supseteq S'\}$,
- $[\![\exists x \in \psi_1(Z_1, \mathcal{L}_1).[\psi_2(Z_2, \mathcal{L}_2)]]\!]_{\mathcal{EI}} = \bigcup\{[\![\psi_2(Z_2, \mathcal{L}_2)]\!]_{\mathcal{EI}_2 \cup \mathcal{I}_x}\colon\ \mathcal{I}_x(x) \in [\![\psi_1(Z_1, \mathcal{L}_1)]\!]_{\mathcal{EI}_1}\}$,
- $[\![\forall x \in \psi_1(Z_1, \mathcal{L}_1).[\psi_2(Z_2, \mathcal{L}_2)]]\!]_{\mathcal{EI}} = \bigcap\{[\![\psi_2(Z_2, \mathcal{L}_2)]\!]_{\mathcal{EI}_2 \cup \mathcal{I}_x}\colon\ \mathcal{I}_x(x) \in [\![\psi_1(Z_1, \mathcal{L}_1)]\!]_{\mathcal{EI}_1}\}$.

Theorem 4.1. *Model checking of Generalized Quantified μ Calculus is* PSPACE*-complete.*

Proof. It has been shown in [7] that the model checking problem for Quantified CTL in its general form is PSPACE-complete. Since Generalized Quantified μ Calculus subsumes Quantified CTL it follows that Generalized Quantified μ Calculus model checking is PSPACE-hard.

To prove that Generalized Quantified μ Calculus is in PSPACE, we see that given an instantiation of each variable to a state of the Kripke structure the task of model checking reduces to the model checking of a μ-calculus formula (including the task of verifying whether the instantiations are from correct domains). It is known that μ-calculus model checking is in NP. The instantiations can be successively generated in polynomial space, and for each instantiation the μ-calculus formula can be checked using the NP algorithm for μ-calculus. It follows that Generalized Quantified μ Calculus model checking is in $\text{PSPACE}^{\text{NP}}$ (a PSPACE Turing Machine with an NP Oracle). Since $\text{PSPACE}^{\text{NP}} \subseteq \text{PSPACE}^{\text{PSPACE}} = \text{PSPACE}$, it follows that Generalized Quantified μ Calculus model checking is in PSPACE. □

Over and above the initial result presented in [7], Theorem 4.1 shows that even though Generalized Quantified μ Calculus is significantly richer than Quantified CTL, the model checking complexity remains PSPACE-complete in the general case.

5. Quantification in linear temporal logic

In this section, we present results on the effect of using first-order quantification over states in Linear Temporal Logic (LTL). It is known that LTL model checking is PSPACE-complete. However, LTL model checking is very popular (and feasible) because the *program complexity* of LTL model checking (namely, the complexity of checking LTL formulas of constant length) is linear in the size of the Kripke structure. In other words, the exponential in LTL model checking algorithms is in the length of the formula (and not in the size of the Kripke structure, which is typically much larger).

In this section, we show that even for a very restricted use of first-order quantification over states the program complexity of LTL model checking becomes intractable. We call this simple extension *Scope Restricted Quantified LTL*. The syntax of this logic is as follows.

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid X(\phi) \mid \phi U \phi \mid \exists x \in p.[\phi'(x)] \mid \forall x \in p.[\phi'(x)] \quad \text{where } p \in \mathcal{AP},$$
$$\phi'(x) ::= x \mid \phi'(x) \wedge \phi'(x) \mid \phi'(x) \wedge \phi \mid \neg\phi'(x) \mid X(\phi'(x)) \mid \phi U \phi'(x).$$

The semantics of this logic is as follows. Given the structure M let $\pi = s_0, s_1, s_2, \ldots$ be a path (an infinite sequence of states) such that $\forall i,\ i \geqslant 0, \mathcal{R}(s_i, s_{i+1})$. For a given path π we use π^i to denote the *suffix* of π starting

at s_i. For a given path π in M and a path formula ϕ we denote $M, \pi \models \phi$ to denote whether the path π satisfies the formula ϕ.

The truth of a subformula $\phi'(x)$ in a path of the Kripke structure is subject to an instantiation of the free variable x. A free variable x can be instantiated to a state s, in which case we label s with x treating x as an atomic proposition true only at s. Therefore, instantiation is a labeling function $\mathcal{I}: \{x\} \to \mathcal{S}$ which labels the free variable x exactly at one state. Given a formula $\phi'(x)$ and an instantiation $\mathcal{I}$ we use $M, \pi \models \phi'(x)_{\mathcal{I}}$ to denote that $\phi'(x)$ is true in π under instantiation $\mathcal{I}$.

- $M, \pi \models p$ iff $p \in \mathcal{F}(s_0)$, where s_0 is the starting state of π,
- $M, \pi \models \neg\phi$ iff $M, \pi \not\models \phi$,
- $M, \pi \models \phi_1 \wedge \phi_2$ iff $M, \pi \models \phi_1$ and $M, \pi \models \phi_2$,
- $M, \pi \models X(\phi)$ iff $M, \pi^1 \models (\phi)$,
- $M, \pi \models \phi_1 U \phi_2$ iff $\exists k \geqslant 0$ such that $M, \pi^k \models \phi_2$ and $\forall j, 0 \leqslant j < k$ $M, \pi^j \models \phi_1$,
- $M, \pi \models x_{\mathcal{I}}$ iff $\mathcal{I}(x) = s_0$ where s_0 is the starting state of π,
- $M, \pi \models \neg\phi'(x)_{\mathcal{I}}$ iff $M, \pi \not\models \phi'(x)_{\mathcal{I}}$,
- $M, \pi \models (\phi'_1(x) \wedge \phi'_2(x))_{\mathcal{I}}$ iff $M, \pi \models \phi'_1(x)_{\mathcal{I}}$ and $M, \pi \models \phi'_2(x)_{\mathcal{I}}$,
- $M, \pi \models (\phi'_1(x) \wedge \phi_2)_{\mathcal{I}}$ iff $M, \pi \models \phi'_1(x)_{\mathcal{I}}$ and $M, \pi \models \phi_2$,
- $M, \pi \models X(\phi'(x))_{\mathcal{I}}$ iff $M, \pi^1 \models (\phi'(x))_{\mathcal{I}}$,
- $M, \pi \models (\phi_1 U \phi'_2(x))_{\mathcal{I}}$ iff $\exists k \geqslant 0$ such that $M, \pi^k \models \phi'_2(x)_{\mathcal{I}}$ and $\forall j, 0 \leqslant j < k$ $M, \pi^j \models \phi_1$,
- $M, \pi \models \exists x \in p.[\phi'(x)]$ iff there exists an instantiation $\mathcal{I}: \{x\} \to S$ such that $p \in \mathcal{F}(\mathcal{I}(x))$ and $M, \pi \models \phi'(x)_{\mathcal{I}}$,
- $M, \pi \models \forall x \in p.[\phi'(x)]$ iff for all instantiations $\mathcal{I}: \{x\} \to S$ where $p \in \mathcal{F}(\mathcal{I}(x))$ we have $M, \pi \models \phi'(x)_{\mathcal{I}}$,
- $M, s \models \phi$ iff for all paths π with starting state s, we have $M, \pi \models \phi$.

We use the abbreviations: $F(f) = \mathit{true}\, U\, f$ (eventually f) and $G(f) = \neg F(\neg f)$ (always f).

Theorem 5.1. *The program complexity of model checking for Scope Restricted Quantified LTL is* NP*-hard and* coNP*-hard.*

Proof. We reduce the Hamiltonian path problem to the model checking problem of Scope Restricted Quantified LTL. Consider the graph $G = \langle V, E\rangle$ for which we are required to find a Hamiltonian path. We translate this graph to a Kripke structure $M = \langle \mathcal{S}, \mathcal{R}, s_0, \mathcal{AP}, \mathcal{F}\rangle$ as follows: $\mathcal{S} = V \cup \{u_0, u_1\}$ such that $u_0, u_1 \notin V$, $\mathcal{R} = E \cup \{\langle u_0, v\rangle\colon\ v \in V\} \cup \{\langle v, u_1\rangle\colon\ v \in V\} \cup \{\langle u_1, u_1\rangle\}$, $s_0 = u_0$, $\mathcal{AP} = \{\mathit{true}, p\}$, and $\mathcal{F}(v) = \{\mathit{true}, p\}$, $\forall v \in V$ and $\mathcal{F}(u_0) = \mathcal{F}(u_1) = \{\mathit{true}\}$.

Let us consider the formula $\psi = \forall x \in p.[F(x) \wedge G(x \Rightarrow XG(\neg x))]$. The path formula $F(x) \wedge G(x \Rightarrow X(G(\neg x)))$ is true on a path if there is a state labeled x on the path (expressed as $F(x)$) and it holds globally that whenever x holds in a state of the path then from the next state onwards x never occurs again ($G(x \Rightarrow X(G(\neg x)))$. Hence it is easy to see that $M, u_0 \models \neg\psi$ iff there is no such path. Hence $M, u_0 \models \neg\psi$ iff the graph G does not have a Hamiltonian path. Thus ψ is a formula of constant length which describes the Hamiltonian path of a graph using scope restricted quantification on path. Hence the program complexity of Scope Restricted Quantified LTL is NP-hard and coNP-hard. □

6. Generalized quantified LTL

Though the program complexity of LTL model checking becomes significantly harder under quantification over states, the overall complexity of LTL model checking remains PSPACE-complete even when we generalize the scope of quantification. We shall refer to the following extension of LTL with unrestricted use of quantification over states as *Generalized Quantified LTL*.

$$\phi(\mathcal{L}) ::= \neg\phi(\mathcal{L}) \mid \phi(\mathcal{L}) \wedge \phi(\mathcal{L}) \mid X(\phi(\mathcal{L})) \mid \phi(\mathcal{L}) U \phi(\mathcal{L}) \mid p \mid y \text{ where } p \in \mathcal{AP} \text{ and } y \in \mathcal{Y},$$
$$Q(\mathcal{L}) ::= \exists x \in Q(\mathcal{L}).[Q(x \parallel \mathcal{L})] \mid \forall x \in Q(\mathcal{L}).[Q(x \parallel \mathcal{L})] \text{ where } x \notin \mathcal{L}$$
$$\mid Q(\mathcal{L}) \wedge Q(\mathcal{L}) \mid X(Q(\mathcal{L})) \mid \neg Q(\mathcal{L}) \mid \phi(\mathcal{L}) \mid Q(\mathcal{L}) U Q(\mathcal{L}) \mid x \text{ where } x \in \mathcal{X}.$$

The roles of the variables and their meanings are similar to that of Section 4. Generalized Quantified LTL formulas are of the form $Q([\,])$ where $[\,]$ is the empty list. The semantics is also similar, however we present the semantics for the interesting ones:

- $M, \pi \models (\psi_1(\mathcal{L}_1) U \psi_2(\mathcal{L}_2))_{\mathcal{I}}$ iff $\exists k \geqslant 0$, $M, \pi^k \models \psi_2(\mathcal{L}_2)_{\mathcal{I}_2}$ and $\forall j$, $0 \leqslant j < k$, $M, \pi^j \models \psi_1(\mathcal{L}_1)_{\mathcal{I}_1}$ where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$).
- $M, s \models \psi_{\mathcal{I}}$ iff for all paths π with starting state s and $M, \pi \models \psi_{\mathcal{I}}$.
- $M, \pi \models \exists x \in \psi_1(\mathcal{L}_1)[\psi_2(\mathcal{L}_2)]$ iff $\exists$ instantiation $\mathcal{I}_x : \{x\} \to \mathcal{S}$ of x such that $M, I_x(x) \models \psi_1(\mathcal{L}_1)_{\mathcal{I}_1}$ and $M, \pi \models \psi_2(\mathcal{L}_2)_{\mathcal{I}_2 \cup \mathcal{I}_x}$, where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$).
- $M, \pi \models \forall x \in \psi_1(\mathcal{L}_1)[\psi_2(\mathcal{L}_2)]$ iff $\forall$ instantiation $\mathcal{I}_x : \{x\} \to \mathcal{S}$ of x such that $M, I_x(x) \models \psi_1(\mathcal{L}_1)_{\mathcal{I}_1}$ we have $M, \pi \models \psi_2(\mathcal{L}_2)_{\mathcal{I}_2 \cup \mathcal{I}_x}$, where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$).

Theorem 6.1. *Model checking Generalized Quantified LTL is* PSPACE*-complete.*

Proof. Since Generalized Quantified LTL subsumes LTL and LTL model checking is PSPACE-complete [8], the model-checking problem for Generalized Quantified LTL is PSPACE-hard.

It follows from the semantics of Generalized Quantified LTL that given an instantiation of each variable the task of model checking reduces to model checking of a LTL formula (including the verification of the correct domains of instantiations). The instantiations can be generated in polynomial space recursively and for each instantiation the LTL formula can be verified in PSPACE using the PSPACE algorithm for model checking a LTL formula [8]. Hence we have that model checking of Generalized Quantified LTL $\in \text{PSPACE}^{\text{PSPACE}}$ (that is, a PSPACE Turing machine with a PSPACE oracle). Hence Generalized Quantified LTL can be checked in PSPACE, since $\text{PSPACE}^{\text{PSPACE}} = \text{PSPACE}$. □

7. Conclusion

In this work, we have shown that the power of first-order quantification over states in branching time and linear time temporal logics comes with an associated computational overhead in terms of model checking complexity. In the case of branching time logics its scope-restricted use is computationally effective as well as enriching in terms of expressibility. In the case of linear temporal logic, the model checking problem remains similar (PSPACE-complete), but the increase in program complexity is likely to limit its applicability to verifying high-level abstractions of state-machines.

Acknowledgements

Pallab Dasgupta and P.P. Chakrabarti acknowledge the support of the Department of Science and Technology, Government of India, for this work.

References

[1] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, Cambridge, MA, 2000.

[2] M. Dam, CTL* and ECTL* as fragments of the modal μ-calculus, Theoret. Comput. Sci. 126 (1994) 77–96.
[3] E.A. Emerson, Model checking and the mu calculus, in: N. Immerman, P. Kolatis (Eds.), Proc. of DIMACS Symposium on Descriptive Complexity and Finite Model, 1996, pp. 185–214.
[4] E.A. Emerson, C.L. Lei, Efficient model checking in fragments of the mu calculus, in: IEEE Symposium on Logic in Computer Science (LICS), Cambridge, MA, June 1986.
[5] D. Kozen, Results on the propositional μ-calculus, Theory Comput. Sci. 27 (1983) 333–354.
[6] O. Kupferman, Augmenting branching temporal logics with existential quantification over atomic propositions, J. Logic Comput. 9 (2) (1999) 135–147.
[7] A.C. Patthak, I. Bhattacharya, A. Dasgupta, P. Dasgupta, P.P. Chakrabarti, Quantified computation tree logic, Inform. Process. Lett. 82 (2002) 123–129.
[8] A.P. Sistla, E.M. Clarke, The complexity of propositional linear temporal logics, J. ACM 32 (3) (1985) 733–749.
[9] A.P. Sistla, M.Y. Vardi, P. Wolper, The complementation problem for Buchi automata with applications to temporal logic, in: 10th ICALP, in: Lecture Notes in Comput. Sci., vol. 194, Springer, Berlin, 1985, pp. 465–474.