ELSEVIER

Information Processing Letters 82 (2002) 123–129

Information Processing Letters

www.elsevier.com/locate/ipl

Quantified Computation Tree Logic

A.C. Patthak, I. Bhattacharya, A. Dasgupta *, Pallab Dasgupta, P.P. Chakrabarti

Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India 721302

Received 4 January 2001; received in revised form 4 July 2001
Communicated by K. Iwama

Abstract

Computation Tree Logic (CTL) is one of the most syntactically elegant and computationally attractive temporal logics for branching time model checking. In this paper, we observe that while CTL can be verified in time polynomial in the size of the state space times the length of the formula, there is a large set of reachability properties which cannot be expressed in CTL, but can still be verified in polynomial time. We present a powerful extension of CTL with first-order quantification over sets of reachable states. The extended logic, QCTL, preserves the syntactic elegance of CTL while enhancing its expressive power significantly. We show that QCTL model checking is PSPACE-complete in general, but has a rich fragment (containing CTL) which can be checked in polynomial time. We show that this fragment is significantly more expressive than CTL while preserving the syntactic beauty of CTL.

Keywords: Computation Tree Logic; Model checking; Verification

1. Introduction

Temporal logic model checking [4,6] has emerged as one of the most powerful techniques for verifying temporal properties of finite state programs [3,11, 14]. In this approach, the program is modeled as a finite state non-deterministic transition system. The correctness property that needs to be verified on the transition system is specified in terms of a temporal logic formula. Model checking has been extensively studied for two broad categories of temporal logics, namely *linear time temporal logic* and *branching time temporal logic* [5].

The branching time temporal logic, *Computation Tree Logic* (CTL) [4], is one of the most popular temporal logics in practice. CTL allows us to express a wide variety of branching time properties which can be verified in polynomial time (that is, the time complexity of CTL model checking is polynomial in the size of the state transition system times the length of the CTL formula). This makes CTL model checking computationally attractive as compared to the linear time temporal logic, LTL, and the more expressive branching time logic CTL*. Model checking with LTL and CTL* are known to be PSPACE-complete [4,6].

Intuitively, CTL model checking derives its computational superiority over temporal logics such as LTL and CTL* from the fact that CTL formulas can be recursively decomposed into subformulas which can be checked independently on the states of the system. Once this is done, the verification of the formula itself reduces to a simple question of reachability through the states marked by the subformulas. Since the reachability question can be answered in polynomial time,

* Corresponding author.
E-mail addresses: pallab@cse.iitkgp.ernet.in (A. Dasgupta), ppchak@cse.iitkgp.ernet.in (P.P. Chakrabarti).

0020-0190/02/$ – see front matter
PII: S0020-0190(01)00260-5

we get a dynamic programming solution to the problem which works in polynomial time [6].

The work described in this paper is inspired by the fact that there exists a wide variety of interesting branching time temporal properties which can be evaluated in polynomial time, but are not expressible in CTL. For example, if we wish to determine whether there exists a path from state s_i to *some* state where the property f is true, we can express this in CTL as $E(true\,U f)$. On the other hand, if we wish to determine whether there exists paths from state s_i to *each* state where the property f is true, we cannot express this succinctly in CTL without explicitly enumerating the states where f is true. Similarly, suppose we wish to determine whether there exists a state t such that q is true at t, there exists a path from s_i to t through states labeled by p, and there exists a path from s_i to t through states labeled by r. This query is not expressible in CTL (the CTL formula $E(p\,U\,q) \wedge E(r\,U\,q)$ does not force us to find paths to a common state t where q is true).

Formally, in a CTL formula of the form $E(f\,U\,g)$ (where f and g are CTL subformulas), we always look for a path to *some* state where g is true through states where f is true, thereby implicitly imposing an existential quantification over the set of states which are labeled by g. In other words, if G denotes the set of states where g is true, then $E(f\,U\,g)$ asks whether $\exists x \in G, E(f\,U\,x)$. If we wished to determine whether there are f-paths to each state in G, we could have written this property as $\varphi = \forall x \in G, E(f\,U\,x)$. This universal quantification over the set G cannot be expressed succinctly in CTL, though it is not difficult to establish that if f and g are polytime checkable, then so is φ.

In this paper, we extend CTL formulas to allow both universal and existential quantification over the sets of states where its subformulas are true. The resulting logic, which we call *Quantified CTL* (QCTL) borrows the quantifiers $\exists$ and $\forall$ from first-order logic. In this context it should be noted that the relation between first-order logic and temporal logics have been extensively studied by researchers [8,13]. It has also been shown that CTL and CTL* can be expressed using well defined fragments of first-order logic [1, 9]. On the other hand, extending the expressibility of CTL while preserving its elegant syntactic structure is attractive from the specificational point of view. The idea of extending the expressive power of CTL by importing first-order quantifiers has been explored in [10] as well, though with a different objective and an altogether different semantics. In [10], the quantification was done over atomic propositions. In contrast we have seen that extending quantification over sets of states which satisfy a sub-formula allows us to express a wide range of queries which are not succinctly expressible in CTL. The logic proposed in this paper preserves the syntactic elegance of CTL while enhancing its expressive power.

In this paper we present the logic QCTL which allows both universal and existential quantification over sets of states satisfying sub-properties. We establish that in its general form, QCTL model checking is PSPACE-complete. We then identify a rich fragment of QCTL by restricting the scope of the quantifications. The restricted fragment of QCTL contains CTL and preserves the syntactic elegance of CTL. We establish that the restriction of QCTL can be verified in time polynomial in the size of the state space times the length of the formula.

The paper is organized as follows. In Section 2 we illustrate the proposed logic through examples. Section 3 presents the formal syntax and semantics of QCTL. In Section 4 we establish that QCTL model checking is PSPACE-complete in general. The polytime checkable fragment of QCTL is presented in Section 5.

2. Examples

In this section we illustrate the proposed logic QCTL through a couple of motivating examples. Consider the transition system shown in Fig. 1. States labeled by the atomic proposition, t, are the *terminal* states of the systems. Similarly states labeled by c are *control* states and states labeled by o are *observable* states.

We now illustrate some interesting properties which can be expressed in QCTL but not succinctly in CTL.

- *Are all terminal states reachable from* s_0? We write this in QCTL as:

  $$\varphi_1 = \forall x \in t \left[E(true\,U\,x)\right].$$

  In Fig. 1, this formula is true at s_0.

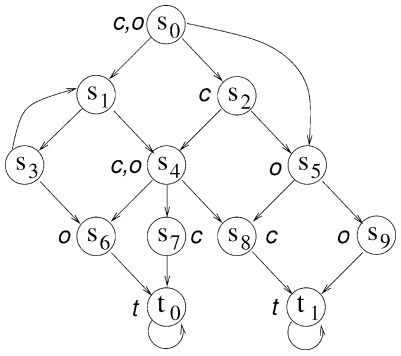

Fig. 1. Sample transition system.

- *Is there a terminal state which can be reached from s_0 through only control states as well as through only observable states*? This is expressed in QCTL as:

  $$\varphi_3 = \exists x \in t\big[E(c\,U\,x) \wedge E(o\,U\,x)\big].$$

  In Fig. 1, this formula is false at s_0, though there is a path through control states to t_0, and a path through observable states to t_1.
- *Is there a state in future which satisfies the previous property, φ_3, and can be reached only through control states from s_0*? This composite property can be expressed as:

  $$\exists x \in (\varphi_3)\big[E(c\,U\,x)\big]$$

  which in its full form is as follows:

  $$\exists x \in \big(\exists y \in t\big[E(c\,U\,y) \wedge E(o\,U\,y)\big]\big)\big[E(c\,U\,x)\big].$$

  Note that x is chosen from the set of states which satisfy φ_3. This property is indeed true at s_0, with φ_3 being true at s_4.
- *Is there a terminal state which is reached along all paths*? We write this as:

  $$\exists x \in t\big[A(\mathit{true}\,U\,x)\big].$$

  This property is false at s_0, but true at s_5.
- *Is it the case that every path to a terminal state reaches the terminal state only through states which are either control states or observables states*? We write this in QCTL as:

  $$\neg\exists x \in t\big[E\big(\mathit{true}\,U\big(\neg(c \vee o) \wedge E(\mathit{true}\,U\,x) \wedge \neg x\big)\big)\big].$$

  This property is false at s_0, but true at s_2.

None of the above properties are expressible in CTL. However, every CTL formula is a QCTL formula without quantifiers. Therefore, QCTL is strictly more expressive than CTL.

QCTL is also useful for specifying interesting graph reachability queries. For example, let us consider the problem of identifying the *kings* in a *tournament* graph [15]. A *tournament* is an orientation[1] of a complete graph (clique). An edge from vertex x_i to vertex x_j indicates that team x_i has won over team x_j. A *king* of the tournament is a vertex x_k such that for every other vertex x_i, either there is an edge from x_k to x_i (that is, x_k defeated x_i) or there is an edge from x_k to some vertex x_j such that x_j has an edge to x_i (that is, x_k has defeated someone who defeated x_i). It is easy to see that the kings of a tournament $G = (V, E)$ is exactly the set of vertices that can be labeled by the following QCTL formula:

$$\varphi = \forall i \in \mathit{true}\big[i \vee EX(i) \vee EXEX(i)\big].$$

If we wanted to determine whether there is any team which has defeated all kings, we could specify that as:

$$\forall k \in \varphi\big[EX(k)\big],$$

where φ is as above.

3. Syntax and semantics of QCTL

In this section we present the formal syntax and semantics of QCTL in its general form. In Section 4 we shall establish that QCTL model checking is PSPACE-complete in general, and in Section 5 we present a rich subset of QCTL which can be checked in polynomial time.

The truth of QCTL formulas are interpreted over a finite Kripke structure, $J = \langle \mathcal{AP}, S, \mathcal{R}, s_0, \mathcal{F}\rangle$, where:

- $\mathcal{AP}$ is a set of atomic propositions,
- S is a finite set of states,
- $\mathcal{R} \subseteq S \times S$ is a transition relation, where $(s_i, s_j) \in \mathcal{R}$ implies that s_j is a successor state of the state s_i,
- $s_0 \in S$ is the initial state,
- $\mathcal{F}: S \to 2^{\mathcal{AP}}$ is a labeling of states with atomic propositions true in that state.

A *path*, π, in the Kripke structure is an infinite sequence of states, $s_0, s_1, \ldots$, such that for all i, $s_i \in S$

[1] An orientation of an undirected graph is a directed graph obtained by giving a direction to its edges.

and $\mathcal{R}(s_i, s_{i+1})$. s_0 is called the starting state of π. Since the Kripke structure has a finite set of states, one or more states will appear multiple number of times on a path. In other words, a path (as defined here) is an infinite *walk* over the state transition graph.

The formal grammar of QCTL is as follows:

$$\begin{aligned} C(\mathcal{L}) = {} & \neg C(\mathcal{L}) \mid C(\mathcal{L}) \wedge C(\mathcal{L}) \mid C(\mathcal{L}) \vee C(\mathcal{L}) \\ & \mid AX\big(C(\mathcal{L})\big) \mid EX\big(C(\mathcal{L})\big) \\ & \mid A\big(C(\mathcal{L})\, U\, C(\mathcal{L})\big) \mid E\big(C(\mathcal{L})\, U\, C(\mathcal{L})\big) \mid p \end{aligned}$$

where $p \in \mathcal{AP} \cup \mathcal{L}$,

$$\begin{aligned} Q(\mathcal{L}) = {} & \exists x \in Q(\mathcal{L})\big[Q(x \parallel \mathcal{L})\big] \quad \text{where } x \notin \mathcal{L} \\ & \mid \forall x \in Q(\mathcal{L})\big[Q(x \parallel \mathcal{L})\big] \quad \text{where } x \notin \mathcal{L} \\ & \mid Q(\mathcal{L}) \wedge Q(\mathcal{L}) \mid Q(\mathcal{L}) \vee Q(\mathcal{L}) \\ & \mid AX\big(Q(\mathcal{L})\big) \mid EX\big(Q(\mathcal{L})\big) \\ & \mid A\big(Q(\mathcal{L})\, U\, Q(\mathcal{L})\big) \mid E\big(Q(\mathcal{L})\, U\, Q(\mathcal{L})\big) \\ & \mid \neg Q(\mathcal{L}) \mid C(\mathcal{L}) \end{aligned}$$

$QCTL = Q([])$ where [] stands for an empty list.

The symbol $\mathcal{L}$ represents a list of *free* variables. $C(\mathcal{L})$ and $Q(\mathcal{L})$ respectively represent CTL and QCTL subformulas with free variables in $\mathcal{L}$. The string $x \parallel \mathcal{L}$ represents the appending of the variable x in the list $\mathcal{L}$. The non-terminal *QCTL* represents QCTL formulas. It should be noted that $QCTL = Q([])$, that is, a QCTL formula cannot have a free variable ($\mathcal{L}$ is empty), though its subformulas may have free variables. $C([])$ represents traditional CTL formulas (without free variables).

The truth of a QCTL sub-formula, $Q(\mathcal{L})$, at a state of the Kripke structure is subject to an *instantiation* of the free variables in its list $\mathcal{L}$. A free variable $x \in \mathcal{L}$ can be instantiated to a state v, in which case we *label* v with x and treat x as an atomic proposition true only in v. Thus an instantiation of $\mathcal{L}$ can be viewed as a labeling function $\mathcal{I}: \mathcal{L} \to S$, where each label $x \in \mathcal{L}$ can be used on at most one state. For QCTL subformulas without free variables, $\mathcal{I}$ is always empty. Given a QCTL subformula $\varphi(\mathcal{L})$ and an instantiation $\mathcal{I}$, we use the notation $s \models_{\mathcal{I}} \varphi(\mathcal{L})$ to indicate that $\varphi(\mathcal{L})$ is true at state s under the instantiation $\mathcal{I}$. If $\mathcal{L}$ is empty, then we use the notation $s \models \varphi$ to indicate that φ is true at s. Likewise if a path formula ψ is true in a path π under instantiation $\mathcal{I}$, we denote this by $\pi \models_{\mathcal{I}} \psi$. The semantics of QCTL is as follows.

- $\forall s \in S, s \models \mathit{True}$ and $s \not\models \mathit{False}$,
- $s \models p$ iff $p \in \mathcal{F}(s)$,
- $s \models_{\mathcal{I}} x$ iff $\mathcal{I}(x) = s$,
- $s \models_{\mathcal{I}} \neg\varphi(\mathcal{L})$ iff $s \not\models_{\mathcal{I}} \varphi(\mathcal{L})$,
- $s \models_{\mathcal{I}} \varphi_1(\mathcal{L}_1) \wedge \varphi_2(\mathcal{L}_2)$ iff $s \models_{\mathcal{I}_1} \varphi_1(\mathcal{L}_1)$ and $s \models_{\mathcal{I}_2} \varphi_2(\mathcal{L}_2)$, where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$),
- $s \models_{\mathcal{I}} \varphi_1(\mathcal{L}_1) \vee \varphi_2(\mathcal{L}_2)$ iff $s \models_{\mathcal{I}_1} \varphi_1(\mathcal{L}_1)$ or $s \models_{\mathcal{I}_2} \varphi_2(\mathcal{L}_2)$, where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$),
- $s \models_{\mathcal{I}} EX\varphi(\mathcal{L})$ iff $\exists s'$, $\mathcal{R}(s, s')$ such that $s' \models_{\mathcal{I}} \varphi(\mathcal{L})$,
- $s \models_{\mathcal{I}} AX\varphi(\mathcal{L})$ iff $\forall s'$, $\mathcal{R}(s, s')$ we have $s' \models_{\mathcal{I}} \varphi(\mathcal{L})$,
- $\pi \models_{\mathcal{I}} \varphi_1(\mathcal{L}_1)\, U\, \varphi_2(\mathcal{L}_2)$ iff there exists a state $t \in \pi$ such that $t \models_{\mathcal{I}_2} \varphi_2(\mathcal{L}_2)$, and for each state s_i preceding t in π, $s_i \models_{\mathcal{I}_1} \varphi_1(\mathcal{L}_1)$, where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$),
- $s \models_{\mathcal{I}} E(\varphi_1(\mathcal{L}_1)\, U\, \varphi_2(\mathcal{L}_2))$ iff there exists a path π starting at s such that $\pi \models_{\mathcal{I}} \varphi_1(\mathcal{L}_1)\, U\, \varphi_2(\mathcal{L}_2)$,
- $s \models_{\mathcal{I}} A(\varphi_1(\mathcal{L}_1)\, U\, \varphi_2(\mathcal{L}_2))$ iff for each path π starting at s we have $\pi \models_{\mathcal{I}} \varphi_1(\mathcal{L}_1)\, U\, \varphi_2(\mathcal{L}_2)$,
- $s \models_{\mathcal{I}} \exists x \in \varphi_1(\mathcal{L}_1)\ [\varphi_2(x \parallel \mathcal{L}_2)]$ iff there exists an instantiation $\mathcal{I}_x : \{x\} \to S$ of x such that $\mathcal{I}_x(x) \models_{\mathcal{I}_1} \varphi_1(\mathcal{L}_1)$ and $s \models_{\mathcal{I}_2 \cup \mathcal{I}_x} \varphi_2(x \parallel \mathcal{L}_2)$, where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$),
- $s \models_{\mathcal{I}} \forall x \in \varphi_1(\mathcal{L}_1)[\varphi_2(x \parallel \mathcal{L}_2)]$ iff for every instantiation $\mathcal{I}_x : \{x\} \to S$ of x such that $\mathcal{I}_x(x) \models_{\mathcal{I}_1} \varphi_1(\mathcal{L}_1)$ we have $s \models_{\mathcal{I}_2 \cup \mathcal{I}_x} \varphi_2(x \parallel \mathcal{L}_2)$, where $\mathcal{I}_1$ (respectively $\mathcal{I}_2$) is $\mathcal{I}$ with domain restricted to $\mathcal{L}_1$ (respectively $\mathcal{L}_2$).

4. Complexity of QCTL model checking

The following theorem establishes that QCTL model checking is PSPACE-complete.

Theorem 4.1. *QCTL model checking is* PSPACE-*complete.*

Proof. We reduce QBF-SAT, that is, the satisfiability of *Quantified Boolean Formulas* (QBFs) to QCTL model checking. QBF-SAT is known to be PSPACE-complete [12].

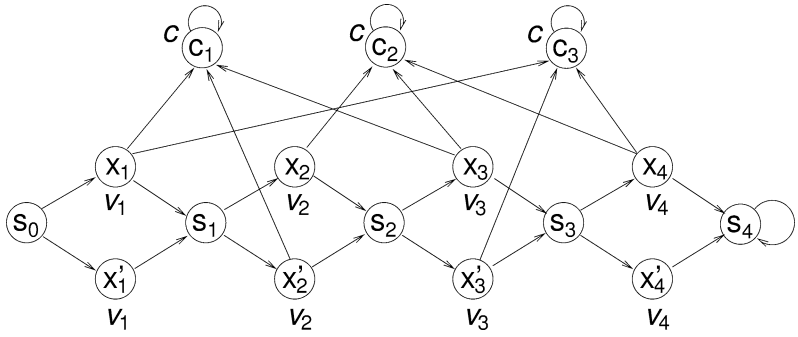

Fig. 2. Model for $\exists x_1 \forall x_2 \exists x_3 \forall x_4 (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee x_4)$.

Given a QBF of the form: $\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n Q(x_1, x_2, \ldots, x_n)$ we illustrate the construction of a Kripke structure, J, and a QCTL formula, ψ, to be verified on the start state of J. Without loss of generality [12], we assume that the Boolean formula $Q(x_1, x_2, \ldots, x_n)$ is in *conjunctive normal form* (CNF) having k clauses $c_1, \ldots, c_k$, where each clause is a disjunction of literals. We construct a Kripke structure: $J = \langle \mathcal{AP}, S, \mathcal{R}, s_0, \mathcal{F} \rangle$, where:

- $\mathcal{AP} = \{V_1, \ldots, V_n\} \cup \{C\}$;
- $S = \{s_0, \ldots, s_n\} \cup \{x_1, \ldots, x_n\} \cup \{x'_1, \ldots, x'_n\} \cup \{c_1, \ldots, c_k\}$;
- $s_0 \in S$ is the initial state;
- the labeling relation $\mathcal{F} : S \to 2^{\mathcal{AP}}$ is as follows:
  - $\forall i, 1 \leqslant i \leqslant n$, $\mathcal{F}(x_i) = \mathcal{F}(x'_i) = V_i$, and
  - $\forall i, 1 \leqslant i \leqslant k$, $\mathcal{F}(c_i) = C$;
- the transition relation $\mathcal{R}$ is as follows:
  - $\forall i, 0 \leqslant i < n$, $\mathcal{R}(s_i, x_{i+1})$ and $\mathcal{R}(s_i, x'_{i+1})$,
  - $\forall i, 0 < i \leqslant n$, $\mathcal{R}(x_i, s_i)$ and $\mathcal{R}(x'_i, s_i)$,
  - $\forall i, 0 < i \leqslant n$ and $\forall j, 0 < j \leqslant k$, $\mathcal{R}(x_i, c_j)$ iff the clause c_j in the given QBF contains the literal x_i,
  - $\forall i, 0 < i \leqslant n$ and $\forall j, 0 < j \leqslant k$, $\mathcal{R}(x'_i, c_j)$ iff the clause c_j in the given QBF contains the literal $\neg x_i$,
  - $\forall j, 0 < j \leqslant k$, $\mathcal{R}(c_j, c_j)$,
  - $\mathcal{R}(s_n, s_n)$.

Fig. 2 shows the Kripke structure corresponding to the following QBF:

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee x_4).$$

The QCTL formula corresponding to this formula will be as follows:

$$\begin{aligned}\exists v_1 \in V_1 \big[\forall v_2 \in V_2 \big[\exists v_3 \in V_3 \big[\forall v_4 \in V_4 \\ \big[\forall c \in C \big[& E(\mathit{true}\, U (v_1 \wedge EX(c))) \vee \\ & E(\mathit{true}\, U (v_2 \wedge EX(c))) \vee \\ & E(\mathit{true}\, U (v_3 \wedge EX(c))) \vee \\ & E(\mathit{true}\, U (v_4 \wedge EX(c)))\big]\big]\big]\big]\big].\end{aligned}$$

In general, the QCTL formula corresponding to the given QBF is of the form:

$$\psi = \exists v_1 \in V_1 \Big[\forall v_2 \in V_2 \Big[\ldots \Big[\forall v_n \in V_n \Big[\forall c \in C \Big[\bigvee_{j \in \{1,n\}} E\big(\mathit{true}\, U (v_j \wedge EX(c))\big) \Big]\Big]\Big] \ldots \Big]\Big].$$

There is a one-to-one correspondence between the instantiations of the variables in the given formula $Q(x_1, \ldots, x_n)$, and the instantiations of the variables in the following QCTL subformula of ψ:

$$\varphi(v_1, \ldots, v_n) = \forall c \in C \Big[\bigvee_{j \in \{1,n\}} E\big(\mathit{true}\, U (v_j \wedge EX(c))\big) \Big].$$

Specifically, if x_i is instantiated to true in Q, then we instantiate v_i to the state x_i, and if x_i is instantiated to false in Q, then we instantiate v_i to the state x'_i, and vice versa. Therefore we require to show that those and only those instantiations which satisfy Q are the ones under which s_0 models φ.

If x_i (respectively x'_i) is chosen by the instantiation, then $E(\mathit{true}\, U (v_i \wedge EX(c)))$ is satisfied only for those clauses, c, which contain x_i (respectively $\neg x_i$) as a

literal (this follows from the definition of the transition relation $\mathcal{R}$). Since we use $\forall c \in C$ to quantify the disjunction of $E(\mathit{true}\,U(v_i \wedge EX(c)))$, we require an instantiation where *each* clause is satisfied by at least one v_i. For the other direction, any instantiation of the v_is for which φ holds at s_0, must satisfy at least one of $E(\mathit{true}\,U(v_i \wedge EX(c)))$ at s_0 for every c. The instantiations of the v_is therefore yields an instantiation of x_is in Q which satisfies Q.

It is easy to see that QCTL is in PSPACE. Given the instantiations of each variable to a state of the Kripke structure, the verification task reduces to CTL model checking (including the verification of whether the instantiations are from the correct domains). This can be done in polynomial space. The instantiations can be generated recursively in polynomial space and verified as above. □

5. An efficiently verifiable fragment of QCTL

Intuitively, the computational complexity of QCTL model checking is dictated by the scope of the quantifiers in the structure of the formula. This is no surprise as it is common in first-order logic. In this section we present a fragment of QCTL which restricts the scope of the quantifiers and yet captures all properties expressible in CTL as well as a wide variety of reachability properties which cannot be expressed in CTL. We show that this fragment of QCTL can be verified in polynomial time.

The grammar of *scope restricted* QCTL is as follows:

$$\begin{aligned} C(x) &= \neg C(x) \mid C(x) \wedge C(x) \mid C(x) \vee C(x) \\ &\quad \mid AX\big(C(x)\big) \mid EX\big(C(x)\big) \\ &\quad \mid A\big(C(x)\,U\,C(x)\big) \mid E\big(C(x)\,U\,C(x)\big) \mid p \end{aligned}$$

where $p \in \mathcal{AP} \cup \{x\}$,

$$\begin{aligned} Q(x) &= \exists y \in Q([])\big[Q(y)\big] \mid \forall y \in Q([])\big[Q(y)\big] \\ &\quad \mid Q(x) \wedge Q(x) \mid Q(x) \vee Q(x) \\ &\quad \mid AX\big(Q(x)\big) \mid EX\big(Q(x)\big) \mid A\big(Q([])\,U\,Q(x)\big) \\ &\quad \mid E\big(Q([])\,U\,Q(x)\big) \mid \neg Q(x) \mid C(x), \end{aligned}$$

$$QCTL = Q([]).$$

It may be noted that in this fragment of QCTL, we restrict subformulas to have at most one free variable, x. $C([])$ represents the set of pure CTL formulas.

All the properties illustrated in Section 2 are contained in this fragment of QCTL. Moreover, the following four basic forms of reachability from a state s, to a set of states Z can be expressed in this fragment.

- *Is there a path from s to some state in Z*? This can be expressed as:

  $\exists z \in Z\big[E(\mathit{true}\,U\,z)\big]$.

  This is equivalent to the CTL formula $E(\mathit{true}\,U\,Z)$.
- *Are there paths from s to every state in Z*? This can be expressed as:

  $\forall z \in Z\big[E(\mathit{true}\,U\,z)\big]$.
- *Is there a state z in Z such that all paths from s reach z*? This can be expressed as:

  $\exists z \in Z\big[A(\mathit{true}\,U\,z)\big]$.
- *For each state z in Z, is it the case that all paths from s reach z*? This can be expressed as:

  $\forall z \in Z\big[A(\mathit{true}\,U\,z)\big]$.

Since Z in turn can denote the set of states labeled by some other property, QCTL generalizes CTL while preserving its decomposable nature. It should be noted that only the first of the above four properties can be expressed succinctly in CTL without enumerating the states in Z. Thus, even the scope restricted fragment of QCTL significantly enhances the expressive power of CTL.

Theorem 5.1. *The truth of a scope restricted QCTL formula f of length $|f|$ can be verified in all states of a Kripke structure $J = \langle \mathcal{AP}, S, \mathcal{R}, s_0, \mathcal{F}\rangle$ in* $O(|f| \cdot |S| \cdot (|\mathcal{R}| + |S|))$ *time.*

Proof. Let us consider a CTL subformula of the form $f = C(x)$. For a given instantiation, $\mathcal{I}$, of the variable x, the problem of verifying the truth of $C(x)$ at a state of the Kripke structure is equivalent to verifying the CTL formula obtained by treating x as an atomic proposition in $C(x)$, which labels only the state $\mathcal{I}(x)$. Using the algorithm presented in [4] we can verify $C(x)$ for any given instantiation of x in $O(|f| \cdot (|\mathcal{R}| + |S|))$ time.

We now use induction on the length of the scope restricted QCTL formula to establish the result. For the basis condition, we have already shown that for a given instantiation of x, a CTL formula $f = C(x)$, can be

verified in $O(|f| \cdot (|\mathcal{R}| + |S|))$ time. Let us now consider a formula of the form $\varphi = \exists y \in Q_1([])\ [Q_2(y)]$. Since each subformula, ψ, of φ of the form $Q([])$ is independent of the instantiation of y, by the induction hypothesis, we can verify ψ at all states in $O(l \cdot |S| \cdot (|\mathcal{R}| + |S|))$ time, where l denotes the length of ψ. If l_1 denotes the length of $Q_1([])$, then verifying $Q_1([])$ at all states requires $O(l_1 \cdot |S| \cdot (|\mathcal{R}| + |S|))$ time.

Once this labeling is done for all subformulas of the form $Q([])$, we can treat these subformulas as atomic propositions labeling the states of the system. The remaining subformulas are CTL subformulas of the form $C(x)$. Thus once the labeling of states with subformulas of the form $Q([])$ is done, $Q_2(y)$ can be looked upon as a CTL formula of the form $C(y)$. For each instantiation of y, we can now verify $Q_2(y)$ at all states in $O(l_2 \cdot (|\mathcal{R}| + |S|))$ time, where $l_2 = |\varphi| - l_1$ denotes the length of $Q_2(y)$. Since there can be at most $|S|$ instantiations of y, the total complexity of verifying $Q_2(y)$ for all instantiations at all states is $O(l_2 \cdot |S| \cdot (|\mathcal{R}| + |S|))$. Combining this with the complexity of verifying $Q_1([])$ at all states, the total complexity of verifying φ at all states is $O(|\varphi| \cdot |S| \cdot (|\mathcal{R}| + |S|))$.

Since both existential and universal quantification may instantiate each state in S, the worst case complexity of verifying a formula of the form $\varphi = \forall y \in Q_1([])\ [Q_2(y)]$ is identical. □

6. Conclusion

This paper proposes the logic QCTL and shows that while QCTL model checking is PSPACE-complete in general, there is a rich fragment of QCTL which can be verified in polynomial time. We have shown that the scope restricted fragment of QCTL is strictly more expressive than CTL. We have also demonstrated (in Section 2) several interesting properties that can be expressed in scope restricted QCTL, but not in CTL.

In recent times, researchers have found the use temporal logics attractive for specifying planning goals [2, 7]. We feel that the logic QCTL has some interesting features which are applicable to planning problems. For example, simple queries such as finding out whether there exists any city in state A which can be reached from a given city in state *B both by an air-route as well as a rail-route*, cannot be expressed in CTL and LTL, but can be expressed in QCTL. Investigating the applicability of QCTL in planning as well as other domains remains an interesting objective.

Acknowledgements

Pallab Dasgupta and P.P. Chakrabarti acknowledge the support of Sun Microsystems, USA, for this work. P.P. Chakrabarti further acknowledges the Department of Science & Technology, Government of India for partial support. Pallab Dasgupta further acknowledges the Indian National Science Academy for partial support.

References

[1] N. Alechina, N. Immerman, Reachability Logic: An efficient fragment of transitive closure logic, Logic J. IGPL 8 (3) (2000) 325–338.
[2] F. Bacchus, F. Kabanza, Planning for temporally extended goals, Ann. of Math. Artificial Intelligence 22 (1998) 5–27.
[3] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: Proc. Workshop on Logic of Programs, Lecture Notes in Comput. Sci., Vol. 131, Springer, Berlin, 1981, pp. 52–71.
[4] E.M. Clarke, E.A. Emerson, A.P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, ACM Trans. Prog. Lang. Systems 8 (2) (1986) 244–263.
[5] E.M. Clarke, R.P. Kurshan, Computer aided verification, IEEE Spectrum 33 (6) (1996) 61–67.
[6] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, Cambridge, MA, 2000.
[7] G. De Giacomo, M.Y. Vardi, Automata-theoretic approach to planning for temporally extended goals, in: Proc. 5th European Conf. on Planning, 1999, pp. 228–240.
[8] N. Immerman, Relational queries computable in polynomial time, Inform. and Control 68 (1986) 86–104.
[9] N. Immerman, M. Vardi, Model checking and transitive closure logic, in: Proc. of CAV'97, 1997, pp. 291–302.
[10] O. Kupferman, Augmenting branching temporal logics with existential quantification over atomic propositions, J. Logic Comput. 9 (2) (1999) 135–147.
[11] O. Lichtenstein, A. Pnueli, Checking that finite state concurrent programs satisfy their linear specification, in: Proc. 12th POPL, 1985, pp. 97–107.
[12] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[13] M. Vardi, Complexity of relational query languages, in: STOC'82, 1982, pp. 137–146.
[14] M. Vardi, Alternating automata and program verification, in: Computer Science Today: Recent Trends and Developments, Lecture Notes in Comput. Sci., Vol. 1000, Springer, Berlin, 1995.
[15] D.B. West, Introduction to Graph Theory, Prentice-Hall, Englewood Cliffs, NJ, 1996.