Quantitative Temporal Reasoning†

E. A. Emerson [1,2] *A. K. Mok* [1] *A. P. Sistla* [3] *J. Srinivasan* [1,4]

1. Department of Computer Sciences,
The University of Texas at Austin,
Austin, Texas 78712 USA

2. Mathematics and Computing Science Department,
Eindhoven University of Technology,
5600MB Eindhoven,The Netherlands

3. Department of Electrical Engineering and Computer Science,
The University of Illinois at Chicago,
Chicago, Illinois 60680 USA

4. Dowell Schlumberger Inc.,
Tulsa, Oklahoma 74134 USA

Abstract

A substantially large class of programs operate in distributed and real-time environments, and an integral part of their correctness specification requires the expression of time-critical properties that relate the occurrence of events of the system. We focus on the formal specification and reasoning about the correctness of such programs. We propose a system of temporal logic, RTCTL (Real-Time Computation Tree Logic), that allows the melding of qualitative temporal assertions together with real-time constraints to permit specification and reasoning at the twin levels of abstraction: qualitative and quantitative. We argue that many practically useful correctness properties of temporal systems, which need to express timing as an essential part of their functionality requirements, can be expressed in RTCTL. We develop a model-checking algorithm for RTCTL whose complexity is linear in the size of the RTCTL specification formula and in the size of the structure. We also present an essentially optimal, exponential time tableau-based decision procedure for the satisfiability of RTCTL formulae. Finally, we consider several variants and extensions of RTCTL for real-time reasoning.

†The work of the first author was supported in part by NSF grant DCR–8511354, ONR URI contract N00014–86–K–0763, and Netherlands NWO grant nf–3/nfb 62–500. The work of the second author was supported in part by ONR Grant number N00014-89-J-1472 and ONR URI contract N00014–86–K–0763. A summary of these results was presented at the Workshop on Automatic Verifiation Methods for Finite State Systems, Grenoble, France, June 12-14, 1989.

1 Introduction

Motivated mainly by the virtue of separating concerns, most research into formal specification and reasoning about correctness of programs has paid little heed to dealing with quantitative temporal properties. In fact, this has proved to be an advantageous abstraction because, in many applications, the correctness properties of a program need to be stated independently of concerns of efficiency, performance, or features (e.g., the speed) of the underlying hardware implementation (cf. (Dijkstra 1976)). Given this, a common characteristic of most temporal or modal logics heretofore proposed for program reasoning (cf. (Pnueli 1977)) is that they provide a formalism for *qualitative* reasoning about change over time. For example, such formalisms allow the expression of assertions such as an event p will *eventually* occur (stated as $F\,p$); note that this assertion places no bound on the time that may elapse before the occurrence of p. Thus, with $p = \mathit{halt}$, $F\,\mathit{halt}$ asserts that a program terminates, and, indeed, such a qualitative sort of correctness temporal property is in fact the strongest one that may be desirable to state of many programs.

On the other hand, there is a substantially large class of programs that operate in distributed and real-time environments including, for example, network communication protocols and embedded real-time control systems, an integral part of whose correctness specification requires the expression of time-critical properties that relate the occurrence of events of the system. For example, consider $p = \mathit{respond}$, in the context of a control system. There, we might want to assert a quantitative correctness property such as $F^{\leq 50}\,\mathit{respond}$, meaning that a response is guaranteed within *bounded* time, namely, 50 time units. By and large, the specification and verification of such systems has been *ad hoc*.

In this paper, we show how to augment temporal logic to handle quantitative assertions in a systematic fashion. We focus primarily on one system of logic, RTCTL (Real-Time Computation Tree Logic), which extends the qualitative temporal logic CTL (Computation Tree Logic, cf. (Emerson and Clarke 1982), (Emerson and Halpern 1982)). Temporal logic in general and CTL in particular have been demonstrated useful to reason about a variety of discrete systems, and thus, an appropriate extension such as RTCTL should naturally allow one to deal with various kinds of real-time applications. RTCTL allows the melding of qualitative temporal assertions together with real-time constraints to permit specification and reasoning at the twin levels of abstraction: qualitative and quantitative. It supports efficient reasoning at both these levels, and permits refinement from the qualitative level down to the quantitative level. Our approach derives power from the fact that standard techniques in temporal logic such as *satisfiability* testing and *model-checking* have been shown to be applicable to automating the construction of, and mechanical reasoning about, concurrent programs (cf. (Emerson and Clarke 1982), (Clarke, Emerson, and Sistla 1983), (Manna and Wolper 1984), (Lichtenstein and Pnueli 1985)). We provide the groundwork for extending these techniques to real-time applications.

The model-checking approach to program verification, proposed in (Clarke and Emerson 1981) and (Clarke, Emerson, and Sistla 1983), may be summarized as follows. The global state transition graph of a finite-state concurrent system may be viewed as a finite temporal structure, and a correctness specification for the system is expressed as a formula in an appropriate propositional temporal logic. The model-checking algorithm is used to determine whether the structure is a model of the formula (i.e., whether the formula is true in the structure), and, thereby, whether the given finite-state program meets a particular correctness specification. This approach is potentially of wide applicability since a large class of concurrent programming problems have finite-state solutions, and the interesting properties of many such systems can be specified in a propositional temporal logic.

The basic idea behind this mechanical model-checking approach to verification of finite-state systems is to make brute force graph reachability analysis efficient and expressive through the use of temporal logic as an assertion language. Of course, much research in protocol verification has attempted to exploit the fact that protocols are frequently finite-state, making exhaustive graph reachability analysis possible. The advantage offered by model-checking seems to be that it provides greater flexibility in formulating specifications through the use of temporal logic as a single, uniform assertion language that can express a wide variety of correctness properties. This makes it possible to reason about, for example, both safety and liveness properties with equal facility.

Because of the simplicity of the model-checking problem and the efficiency of its solution, the model-checking approach has found several applications to the automatic verification of temporal systems. So far, model-checking algorithms for several temporal logics have been used to verify a large number of finite-state systems ranging from examples of concurrent programs presented in the academic literature (such as solutions to the mutual exclusion and other synchronization problem) to network communication protocols to VLSI circuits (cf. (Clarke, Emerson, and Sistla 1983), (Lichtenstein and Pnueli 1985), (Emerson and Lei 1985), (Browne 1986), (Queille and Sifakis 1981), (Clarke and Grumberg 1987)).

We develop a model-checking algorithm for RTCTL which, like the algorithm for CTL, has complexity linear in the size of the RTCTL specification formula and in the size of the global state-space graph. The key observation that makes this possible is that the model-checking algorithm can easily recover not only whether an eventuality is fulfilled, but also when (cf. (Clarke and Emerson 1981), (Emerson and Lei 1986), (Emerson and Lei 1987)).

Next, we focus on the satisfiability problem for RTCTL. We exhibit an exponential time decision procedure for RTCTL, using a tableau-based approach (cf. (Emerson and Clarke 1982), (Emerson and Halpern 1982)), and show that it is essentially optimal. Our algorithm provides a basis for automating the synthesis of programs with timing-constraints by a method similar to the qualitative approach described in (Emerson and Clarke 1982), (Manna and Wolper 1984), and (Pnueli and Rosner 1989).

Finally, we consider some extensions and variants of RTCTL, which allow various combinations of modalities such as $F^{\geq k}q$ and $F^{=k}q$. Somewhat surprisingly, the presence of the $F^{=k}$ modalities causes a blowup in the complexity of satisfiability to be double exponential time complete, while model checking can still be done in polynomial time. The satisfiability decision procedure for the logic with $F^{=k}$ is of complexity deterministic single-exponential in the temporal portion of the input formula, and deterministic double-exponential in the binary representation of the timing constants in it.

The rest of this paper is organized as follows. In Section 2, we present the logic RTCTL and some useful assertions expressible in it. Section 3 deals with real-time model-checking, and Section 4 with the satisfiability problem for RTCTL. Section 5 considers various other quantitative temporal logics derived from CTL. Finally, related work is discussed in the concluding Section 6.

2 The Logic RTCTL

The system of branching time temporal logic CTL (Computation Tree Logic) has been extensively used to specify and reason about correctness properties of concurrent programs (cf. (Emerson and Clarke 1982), (Emerson and Halpern 1982), (Clarke, Emerson, and Sistla 1983)). One disadvantage of CTL and other extant temporal logics, however, is that they lack the ability to express properties of programs related to real-time. In this section, we define RTCTL (Real-Time CTL), an extension to CTL that permits reasoning about time-critical correctness properties of programs, and give a sample of the kinds of program properties RTCTL can express. We begin, however, with a formal definition of the syntax and semantics of CTL.

Let Σ be an underlying alphabet of atomic propositions P, Q, etc. The set of CTL (Computation Tree Logic) formulae is generated by the following rules:

S1. Each atomic proposition P is a formula.

S2. If p, q are formulae, then so are $p \wedge q$ and $\neg p$.

S3. If p, q are formulae, then so are $A(p\,U\,q)$, $E(p\,U\,q)$, and EXp.

A formula of CTL is interpreted with respect to a temporal structure $M = (S, R, L)$ where S is a set of states, R is a binary relation on S that is total (so each state has at least one successor), and L is a labelling which assigns to each state a set of atomic propositions, those intended to be true at the state. Intuitively, the states of a structure could be thought of as corresponding to the states of a concurrent program, the state transitions of which are specified by the binary relation R. A ***fullpath*** $x = s_0, s_1, s_2, \ldots$ in M is an infinite sequence of states such that $(s_i, s_{i+1}) \in R$ for each i; intuitively, a fullpath captures the notion of an execution sequence. We write $M, s \models p$ to mean that "formula p is true at state s in structure M". When M is understood we write only $s \models p$. We define $\models$ by induction on formula structure:

S1. $s_0 \models P$ iff P is an element of $L(s_0)$

S2. $s_0 \models p \wedge q$ iff $s_0 \models p$ and $s_0 \models q$
$s_0 \models \neg p$ iff it is not the case that $s_0 \models p$

S3. $s_0 \models A(p\, U\, q)$ iff for all fullpaths $s_0, s_1, s_2, \ldots$ in M, $\exists i \geq 0$ such that $s_i \models q$ and $\forall j, 0 \leq j < i, s_j \models p$
$s_0 \models E(p\, U\, q)$ iff for some fullpath $s_0, s_1, s_2, \ldots$ in M, $\exists i \geq 0$ such that $s_i \models q$ and $\forall j, 0 \leq j < i, s_j \models p$
$s_0 \models EXp$ iff there exists an R-successor t of s_0 such that $t \models p$

The other propositional connectives are defined as abbreviations in the usual way: $\vee$ for disjunction, $\Rightarrow$ for implication, and $\equiv$ for logical equivalence. Other basic modalities of CTL are also defined as abbreviations: AFq abbreviates $A(\textit{true}\ U\, q)$, EFq abbreviates $E(\textit{true}\ U\, q)$, AGq abbreviates $\neg EF\neg q$, EGq abbreviates $\neg AF\neg q$, and AXq abbreviates $\neg EX\neg q$.

We use $|p|$ to denote the length of formula p. If p is a CTL formula, we take $|p|$ to be the number of nodes in the syntax diagram for p. If p is a (quantitative) RTCTL formula, $|p| = |p'| + c$ where $|p'|$ is the length of the qualitative CTL formula p' obtained from p by deleting time bounds and c is the sum of the length of the bit strings representing in binary the constants in time bounds of p. For example, if $p = AF^{\leq 5}(Q \wedge R)$, then the corresponding qualitative formula p' is $AF(Q \wedge R)$. We have $|p'| = 4$, charging 1 for the modality AF plus 1 for the connective $\wedge$ plus 1 for the proposition q plus 1 for the proposition R. The constant 5 in the time bound is represented by 101 in binary, a bit string of length 3. Thus $|p| = 4 + 3 = 7$.

We now consider some examples of CTL formulae useful to describe qualitative temporal properties of programs. $AF\,q$, for example, specifies the ***inevitability*** of q: q must eventually hold along all paths. Thus, $AG(p \Rightarrow AF\,q)$ says that p ***inevitably leads-to*** q: q eventually holds along every path stemming from a state at which p is true. Similarly, $EF\,q$ indicates that q could ***potentially*** become true: it is true along some one fullpath. Note that none of these modalities allows one to express that q will in fact become true within a certain number, say 10, of state transitions: they merely assert that q will eventually become true.

So we extend CTL to RTCTL. The set of RTCTL (Real-Time Computation Tree Logic) formulae is generated by the rules S1–S3 above together with the rule:

S4. If p, q are formulae and k is any natural number, then $A(p\, U^{\leq k}\, q)$ and $E(p\, U^{\leq k}\, q)$ are formulae.

The temporal structures over which RTCTL formulae are interpreted are the same as CTL structures. The semantics of the new RTCTL modalities are given by:

S4. $s_0 \models A(p\, U^{\leq k}\, q)$ iff for all fullpaths $s_0, s_1, s_2, \ldots$ in M, $\exists i, 0 \leq i \leq k$, such that $s_i \models q$ and $\forall j, 0 \leq j < i, s_j \models p$
$s_0 \models E(p\, U^{\leq k}\, q)$ iff for some fullpath $s_0, s_1, s_2, \ldots$ in M, $\exists i, 0 \leq i \leq k$, such that $s_i \models q$ and $\forall j, 0 \leq j < i, s_j \models p$

Intuitively, k corresponds to the maximum number of permitted transitions along a path of a structure before the eventuality $p\, U\, q$ holds. We follow the convention that each transition takes unit time for execution (but see the remark near the end of Section 3), so k specifies a time bound.

Some other basic modalities of RTCTL are defined as abbreviations: $AF^{\leq k}\, q$ abbreviates $A(\textit{true}\ U^{\leq k}\, q)$ and $EF^{\leq k}\, q$ abbreviates $E(\textit{true}\ U^{\leq k}\, q)$. We also define the modality $G^{\leq k}$ (for each natural number k) as the dual of $F^{\leq k}$, i.e., $AG^{\leq k}\, p$ abbreviates $\neg EF^{\leq k}\, \neg p$, and $EG^{\leq k}\, p$ abbreviates $\neg AF^{\leq k}\, \neg p$.

It is worth pointing that the RTCTL modalities elegantly generalize the analogous CTL ones. Specifically, note that $A(p\, U\, q)$ abbreviates $\exists k : A(p\, U^{\leq k}\, q)$, and, similarly, $E(p\, U\, q)$ abbreviates $\exists k : E(p\, U^{\leq k}\, q)$. This motivates the following definition: If $A(p\, U\, q)$ is true at a state s of an RTCTL structure, we define the ***rank*** of $A(p\, U\, q)$ at s as the smallest natural number k such that $A(p\, U^{\leq k}\, q)$ holds at s. The rank of $AF\,q$, $E(p\, U\, q)$, and $EF\,q$ are defined similarly.

As usual, an RTCTL formula is said to be ***valid*** if it holds at all states of all structures. From the semantics above, it is easy to verify that the RTCTL formulae $A(p\, U^{\leq k}\, q) \equiv (q \vee (p \wedge AX\, A(p\, U^{\leq (k-1)}\, q)))$ and $E(p\, U^{\leq k}\, q) \equiv$

$(q \vee (p \wedge EXE(p\ U^{\leq (k-1)}\ q)))$ are valid for $k \geq 1$. Also $A(p\ U^{\leq 0}\ q) \equiv q \equiv E(p\ U^{\leq 0}\ q)$ is valid. These formulae may be regarded as the analogues of the fixpoint characterizations of the CTL modalities AU and EU ((Emerson and Halpern 1982)).

Note that we do not treat true parallelism but model concurrency in terms of interleaving in the usual way. We should point out that interleaving is a realistic assumption for many real-time applications such as avionics and other on-board embedded systems. For such systems, hardware is typically quite limited. Often there is only (in effect[‡]) a single processor on which many sequential processes are multiprogrammed, i.e., conceptually running concurrently when viewed at a large level of granularity, but actually excuted by interleaving individual instructions from each process on the single processor.

We conclude this section by illustrating how the basic RTCTL modalities could be used to express important correctness properties of programs that must place an explicit bound on the time between events. First, observe that $AF^{\leq k}\ q$, for example, specifies the *bounded inevitability* of q, i.e., q must hold within k steps along all fullpaths. Thus, the RTCTL formula $AG(p \Rightarrow AF^{\leq k}\ q)$ specifies that p always leads-to q within a bounded period of time, viz., k time units. This formula is therefore useful to specify, for example, that a system must respond (with the action q) to an environmental stimulus p within k units of time; the importance of specifications of this kind for temporal systems is underscored in (Jahanian and Mok 1987).

As a second example, consider a family of m processes, the schedules of which are required to satisfy the property of *k-bounded fairness*, i.e., each process should be scheduled for execution at least once every k steps of the system. This can be expressed by the RTCTL formula $\bigwedge_{i=1}^{m} AF^{\leq k} P_i \wedge \bigwedge_{i=1}^{m} AG(P_i \Rightarrow AXAF^{\leq (k-1)} P_i)$, where P_i indicates that process i is executed. The first set of conjuncts ensures that each process is in fact executed along the first k steps, and the AG conjuncts ensure that, once executed, a process must be scheduled for execution again within k steps. We may remark, as an aside, that the property that there be at least one k-bounded fair execution sequence is expressed by the formula $E\big(\bigwedge_{i=1}^{m} F^{\leq k} P_i \wedge \bigwedge_{i=1}^{m} G(P_i \Rightarrow XF^{\leq (k-1)} P_i)\big)$, which does not conform to the syntax of RTCTL.

As a third and final example, consider a system specification that requires that, on sensing an alarm, all normal processes be suspended, and a vigilant mode be entered for at least the next k time units during which only a restricted set of critical activities is performed. The RTCTL formula $AG(\mathit{alarm} \Rightarrow AG^{\leq k} \mathit{vigilant})$ expresses this requirement.

3 Real-Time Model-Checking

In this section, we present an efficient algorithm for the model-checking problem for RTCTL that is of complexity linear in both the size of the structure being checked and the length of the formula. The capability of RTCTL to allow one to reason quantitatively about time in addition to the qualitative reasoning afforded by CTL enhances the utility of model checking for such applications, as timing constraints play a key role in both network protocols and hardware circuits, as well as numerous other real time systems.

Formally, the model-checking problem for RTCTL may be stated as: *Given an* RTCTL *formula* p_0 *and a finite temporal structure* $M = (S, R, L)$, *for some state* $s \in S$ *is it the case that* $M, s \models p_0$*?* (Recall that the RTCTL structure is said to be finite if its *size*, $|M|$, defined as $|S| + |R|$, is finite.)

Figure 1 presents the overall algorithm for this problem. It determines, for each state s in M and for each subformula p of p_0, whether $M, s \models p$. The algorithm is designed to operate in stages: the first stage processes all subformulae of p_0 of length 1, the second, of length 2, and so on. At the end of the ith stage, each state is labelled with the set of all subformulae of p_0 of length no more than i that are true at the state. As the basis, note that the labelling L of M initially contains the set of atomic propositions (i.e., all subformulae of p_0 of length 1) true at each state of M. To perform the labelling on subsequent iterations, information gathered in earlier iterations is used. For example, a subformula of the form $q \wedge r$, i.e., one whose main connective is $\wedge$, should be added to the labels of precisely those states already labelled with both q and r. Subformulae of the form $\neg q$ are handled in like fashion.

[‡]Hardware duplication may be used for the sake of fault tolerance.

```
/* Input:  A structure M = (S, R, L) and an RTCTL formula p0.   */
/* Output: There is a state s ∈ S such that M, s |= p0.          */

for i := 1 to length(p0) do begin
    for each subformula p of p0 of length i do begin
        case structure of p is of the form
            P, an atomic proposition : /* Nothing to do as states of M already labelled with propositions. */;
            q ∧ r        :  for each s ∈ S do
                                if q ∈ L(s) and r ∈ L(s) then add q ∧ r to L(s);
            ¬q           :  for each s ∈ S do
                                if q ∉ L(s) then add ¬q to L(s);
            EX q         :  for each s ∈ S do
                                if q ∈ L(t) for some R-successor t of s then add EX q to L(s);
            A(q U≤k r)   :  AU_check (q, r, min(k, |S|), A(q U≤k r) );
            A(q U r)     :  AU_check (q, r, |S|, A(q U r) );
            E(q U≤k r)   :  EU_check (q, r, min(k, |S|), E(q U≤k r) );
            E(q U r)     :  EU_check (q, r, |S|, E(q U r) );
        end; /* case */
    end;     /* for */
end;         /* for */
if p0 ∈ L(s) for some s ∈ S then Output (true)
                             else Output (false);
```

Figure 1: A Model-Checking Algorithm for RTCTL.

```
procedure AU_check (q, r, k, f);

begin
    for each t ∈ S, count(s) := 0;
    AU^0_Set := {s ∈ S : r ∈ L(s)};
    j := 0;
    while j < k do
        TEMP := AU^j_Set;
        while TEMP ≠ ∅
            remove some t from TEMP;
            for each R-predecessor s of t do
                count(s) := count(s) + 1;
                if ¬ranked(s) and count(s) = degree(s) and q ∈ L(s) then
                    add s to AU^{j+1}_Set;
                    add f to L(s);
            end
        j := j + 1;
    end
end; /* AU_check */
```

Figure 2: The procedure AU_check.

```
procedure EU_check (q, r, k, f);

begin
    EU^0_Set := {s ∈ S : r ∈ L(s)};
    j := 0;
    while j < k do
        TEMP := EU^j_Set;
        while TEMP ≠ ∅
            remove some t from TEMP;
            for each R-predecessor s of t do
                if ¬ranked(s) and q ∈ L(s) then
                    add s to EU^{j+1}_Set;
                    add f to L(s);
            end
        j := j + 1;
    end
end;  /* EU_check */
```

Figure 3: The procedure EU_check.

Modal subformulae are handled by the procedures shown in Figures 2 and 3 as described below. When the algorithm terminates, the label of each state indicates the subformulae of p_0 true at the state.

The modal subformula $A(q\ U^{\leq k}\ r)$, is processed by invoking the *AU_check* procedure. The first two parameters are q and r while the third parameter corresponds to k. Note, however, that for any integer $k \geq |S|$ the subformula $A(q\ U^{\leq k}\ r)$ is true in a particular state iff the subformula $A(q\ U^{\leq |S|}\ r)$ is true in that state, since any path through M involving more than $|S|$ states must form a cycle. For this reason the procedure *AU_check* is invoked with $min(k, |S|)$ as the third argument. Similarly, the procedures for the other modal subformulae are invoked with third argument. at most $|S|$. The fourth parameter is the modal subformula itself.

Correct operation of the procedure *AU_check* can be understood as follows. To determine the states at which $A(q\ U^{\leq k}\ r)$ holds, it computes the rank j of $A(q\ U^{\leq k}\ r)$ at each state in the structure. Recall that the rank of $A(q\ U^{\leq k}\ r)$ at s is the least value of j such that $A(q\ U^{\leq j}\ r)$ holds at s. The algorithm successively computes $AU^{\leq j}$-Set = the set of states of rank j, for each j from 0 to $|S|$. The key idea is that $AU^{\leq j+1} - Set$ = the set of unranked states which are labelled with q and which have all successor states already ranked and in $AU^{\leq j'} - Set$ for some $j' \leq j$.

We point out that the rank of a node is 1 + the maximum of the ranks of its successors, which need not all be equal. We wish the algorithm to assign a rank to a node as soon as all its successors have been ranked. It can test if state s is ranked, by checking if $L(s)$ contains any AU entry of the form $A(qU^{\leq j}r)$ for some j. The algorithm also uses an integer valued counter for each node s, $count(s)$, to keep track of how many successors of s have been ranked. When s is unranked and $count(s) = degree(s)$, then s should be assigned the rank of 1 + the maximum rank of all its successors. However, the algorithm does not have to go back and inspect the ranks of all successors to compute their maximum; nor, does it have to maintain the maximum of all ranked successors as it proceeds. Instead, note that the algorithm assigns ranks to nodes in (nondecreasing) order of rank. Hence, if t is the successor of s that causes s to be assigned a rank then all other successors t' of s have already been assigned ranks $\leq rank(t) = j$. Thus, the algorithm correctly sets the rank of $s = 1 + j = 1 +$ maximum of the ranks of *all* successors of s. Then the original formula $A(q\ U^{\leq k}\ r)$ is added to $L(s)$.

To analyze the complexity of the procedure *AU_check*, we argue as follows. First, from the correctness of the algorithm, we see that the sets AU^j_Set for $j = 0, 1, ..., k$ are pairwise disjoint and are all contained in S. As a consequence the total number of times the body of the inner while loop is executed is bounded by $|S|$. The body of the inner for loop is executed at most once for each element in R. All the initialization and the other statements in the outer while loop are executed at most $|S|$ times. Thus, we see that the complexity of executing the entire procedure *AU_check* is $O(|S| + |R|)$.

We can easily modify the procedure *AU_check* so that we only use two set variables to keep the elements of AU^j_Set and AU^{j+1}_Set for each iteration of the outer while loop instead of using k such variables.

To handle the modality $E(pU^{\leq k}q)$, we use the procedure *EU_check*. The key idea is that $EU^{\leq j+1} - Set =$ the set of unranked states which are labelled with q and which have some successor state already ranked and in $EU^{\leq j'} - Set$ for some $j' \leq j$. The procedure *EU_check* can be proven to be correct by a similar argument and can also be shown to have complexity $O(|S| + |R|)$.

Theorem 1 *The model-checking problem for* RTCTL *is decidable in time linear in both the size of the input structure and the length of the input formula.* $\Box$

Remark 1: It should be emphasized that time bounds larger than the size of the state space need not be computed, since, e.g., $AF^{\leq |S|}q \Rightarrow AF^{\leq k}q$ for any $k > |S|$. Thus, our algorithm does not require cost exponential in the size of the time bounds (cf. (Alur, Courcoubetis, and Dill 1990)).

Remark 2: We should also point out that, with minor modifications to the procedures *AU_check* and *EU_check*, the above algorithm can as efficiently handle more general temporal structures, ones in which each element of the binary relation R is labelled with a nonnegative integer *cost* that intuitively corresponds to the amount of time taken to execute that transition. RTCTL structures as defined in the previous section may be thought of as labelling each element of R with a single unit of time.

4 Satisfiability for RTCTL

We now turn to the problem of determining the satisfiability of an RTCTL formula. This problem may be stated as: *Given an* RTCTL *formula f, is there a temporal structure M and a state s of M such that $M, s \models f$?* If so, M is said to be a *model* of f. The satisfiability problem for temporal logics has been shown to have applications to synthesis of concurrent programs from their temporal specifications (cf. (Emerson and Clarke 1982), (Manna and Wolper 1984), (Pnueli and Rosner 1989)). Note also that the RTCTL formula f is satisfiable iff $\neg f$ is not valid; hence exhibiting a decision procedure for satisfiability amounts to deciding the validity problem (i.e., determining if a given RTCTL formula is valid) as well.

We will give a tableau-based algorithm to decide the satisfiability of RTCTL formulae. A naive way to do this is to translate the given RTCTL formula, f, to an equivalent CTL one, g, by using the fixpoint characterizations of the $AU^{\leq k}$ and $EU^{\leq k}$ modalities to expand each occurrence of these modalities in f. The tableau-based decision procedure for CTL could then be used to determine the satisfiability of g. But the complexity of such an algorithm would be double exponential in $|f|$, as $|g|$ itself would be exponential in $|f|$, and the CTL decision procedure is exponential in the length of its input.

Somewhat surprisingly, we can give a tableau-based decision procedure for RTCTL, whose complexity is still only single exponential in the size of its input, similar to that for CTL (cf. (Emerson and Clarke 1982), (Emerson and Halpern 1982))[§]). Let f be the RTCTL formula whose satisfiability needs to be determined. We first define several useful notions used in the description of the procedure, beginning with the *Fischer-Ladner closure*, $CL(f)$, of an RTCTL formula f (cf. (Fischer and Ladner 1979), (Emerson and Halpern 1982), (Lichtenstein, Pnueli, and Zuck 1985)). For conciseness of presentation, we assume that f does not have any of the abbreviations listed in Section 2 except AXp (for $\neg EX\neg p$). Identifying $\neg\neg p$ with p, and $A(p\ U^{\leq 0}q)$ and $E(p\ U^{\leq 0}q)$ with q for any RTCTL formulae p and q, $CL(f)$ is the smallest set of formulae containing f and satisfying the following eight conditions:

[§] We refer the reader to (Emerson 1990) for a survey of temporal logic decision procedures.

A. $\neg p \in CL(f) \quad \Leftrightarrow \quad p \in CL(f)$,
B. $p \wedge q \in CL(f) \quad \Rightarrow \quad p, q \in CL(f)$,
C. $EX\, p \in CL(f) \quad \Rightarrow \quad p \in CL(f)$,
D. $AX\, p \in CL(f) \quad \Rightarrow \quad p \in CL(f)$,
E. $A(p\, U\, q) \in CL(f) \quad \Rightarrow \quad p,\, q,\, AXA(p\, U\, q) \in CL(f)$,
F. $E(p\, U\, q) \in CL(f) \quad \Rightarrow \quad p,\, q,\, EXE(p\, U\, q) \in CL(f)$,
G. $A(p\, U^{\leq k}\, q) \in CL(f) \Rightarrow \quad p,\, q,\, AXA(p\, U^{\leq (k-1)}\, q) \in CL(f)$ for $k \geq 1$, and
H. $E(p\, U^{\leq k}\, q) \in CL(f) \Rightarrow \quad p,\, q,\, EXE(p\, U^{\leq (k-1)}\, q) \in CL(f)$ for $k \geq 1$.

Note that the size of $CL(f)$ is exponential in $|f|$. We shall call a formula in $CL(f)$ *elementary* if it is of the form $EX\, p$ or $AX\, p$. We define a subset S of $CL(f)$ to be *maximally consistent* iff S satisfies all the following conditions:

1. For each $p \in CL(f)$, $\neg p \in S \Leftrightarrow p \notin S$,
2. $p \wedge q \in S \quad \Leftrightarrow \quad p, q \in S$,
3. $A(p\, U\, q) \in S \quad \Leftrightarrow \quad q \in S$ or $p,\, AXA(p\, U\, q) \in S$,
4. $E(p\, U\, q) \in S \quad \Leftrightarrow \quad q \in S$ or $p,\, EXE(p\, U\, q) \in S$,
5. $A(p\, U^{\leq k}\, q) \in S \quad \Leftrightarrow \quad q \in S$ or $p,\, AXA(p\, U^{\leq (k-1)}\, q) \in S$ for $k \geq 1$,
6. $E(p\, U^{\leq k}\, q) \in S \quad \Leftrightarrow \quad q \in S$ or $p,\, EXE(p\, U^{\leq (k-1)}\, q) \in S$ for $k \geq 1$,
7. $A(p\, U^{\leq 0}\, q) \in S \quad \Leftrightarrow \quad q \in S$,
8. $E(p\, U^{\leq 0}\, q) \in S \quad \Leftrightarrow \quad q \in S$,
9. $A(p\, U^{\leq k}\, q) \in S \quad \Rightarrow$ for all $j \geq k$ such that $A(p\, U^{\leq j}\, q) \in CL(f)$, $A(p\, U^{\leq j}\, q) \in S$, and
10. $E(p\, U^{\leq k}\, q) \in S \quad \Rightarrow$ for all $j \geq k$ such that $E(p\, U^{\leq j}\, q) \in CL(f)$, $E(p\, U^{\leq j}\, q) \in S$.

We now show that the number of maximally consistent subsets of f is only exponential in $|f|$. An *eventuality* is any formula of the form $A(p\, U\, q)$, $E(p\, U\, q)$, $A(p\, U^{\leq k}\, q)$, or $E(p\, U^{\leq k}\, q)$. We shall call a formula in $CL(f)$ *quantitative* if it is of the form $A(p\, U^{\leq k}\, q)$, $AXA(p\, U^{\leq k}\, q)$, $E(p\, U^{\leq k}\, q)$, or $EXE(p\, U^{\leq k}\, q)$. We let $\mathcal{H}$ denote the set of quantitative eventualities that appear in f as subformulae. We can decompose the positive formulae (i.e., formulae not of the form $\neg p$) in $CL(f)$ into $|\mathcal{H}| + 1$ sets: each quantitative eventuality $H = A(p\, U^{\leq k_H}\, q)$ (respectively, $H = E(p\, U^{\leq k_H}\, q)$) that appears in f has a corresponding set, Y_H, which contains all formulae in $CL(f)$ of the form $A(p\, U^{\leq j}\, q)$ and $AXA(p\, U^{\leq j}\, q)$ (respectively, $E(p\, U^{\leq j}\, q)$ and $EXE(p\, U^{\leq j}\, q)$), where $j \leq k_H$. All other positive formulae in $CL(f)$ are members of a separate set, Y_0. It is easy to see that $|Y_0|$ is linear in $|f|$, whereas, for any $H \in \mathcal{H}$, $|Y_H|$ is exponential in the number of bits in k_H, and hence, exponential in $|f|$.

Next, we note from Rule 1 above that in constructing any maximally consistent set S, we have two choices for each formula in Y_0: either include it in S or include its negation in S. For the formulae in Y_H, however, Rules 9 and 10 imply that we can effectively choose only one j, viz., the smallest one, which is no more than k_H, such that $A(p\, U^{\leq j}\, q)$ or $E(p\, U^{\leq j}\, q)$ is in S. Also, once this choice of the smallest j is made, Rules 5 and 6 determine the quantitative elementary formulae of H that must appear in S. Thus, the number of distinct maximally consistent sets is of the order of $2^{|Y_0|} \times \prod_{H \in \mathcal{H}} (k_H + 2)$, i.e., of size $2^{O(|f|)}$. Note, also, that the number of elements in a maximally consistent set is also exponential in $|f|$.

The first step in the decision procedure is to construct the *tableau* for f. This is a directed graph which encodes potential models of f, so that f is satisfiable iff there is a "collapsed" model of f contained in the tableau. The initial tableau, which we denote by T_0, is a directed graph, whose nodes correspond to the maximally consistent sets of $CL(f)$. A node corresponding to the set S is labelled with the formulae in S. We use the elementary formulae in a node to guide us in determining the edges of T_0. An edge is added from node V to node W iff (a) for every formula of the form $AX\, p$ in V, p is in W, and (b) for every formula of the form $\neg EX\, p$ in V, $\neg p$ is in W.

The next step is to prune T_0 by deleting nodes for which the conjunction of the formulae in their label cannot ever label any state of any temporal structure. Despite the fact that RTCTL has more kinds of eventualities than CTL, this step is the same as the pruning step for CTL (cf. (Emerson and Halpern 1982)). The main task of the pruning step in the CTL algorithm is to check for each eventuality in the label of each node, that the eventuality is *fulfilled*. For example, if the qualitative eventuality AFq appears in the label of node s, it must be that along each fullpath from s there does indeed occur a node labelled with q. This is necessary to avoid the problem of "indefinite postponement" where there is a fullpath starting at s all of whose nodes are labelled with AFq and $AXAFq$ but not q. In pruning for such qualitative eventualities, it is necessary and sufficient to check whether there is a Directed Acyclic sub-Graph (DAG) contained in the tableau for that eventuality rooted at that node which certifies fulfillment of that eventuality at that node. For AFq this would be a DAG all of whose frontier nodes are labelled with q.

For RTCTL, it might appear that such a DAG would need to be detected for the quantitative eventualities in the label of a node as well; however, this is not required because the local structure of the tableau (i.e., the way the initial tableau is constructed) guarantees that such a DAG can always be found. The essential point is that the bound on the quantitative eventualities prevents indefinite postponement, because the local structure of the tableau propagates a decremented bound to successor states. For example, if $AF^{\leq 3}q$ labels s but q does not, then the successors of s must be labelled with $AF^{\leq 2}q$, and so on, ensuring fulfillment within 3 steps.

The pruning can thus be performed in time polynomial in the (exponential) size of the tableau. The remainder of the decison procedure is just as for CTL. The formula f is satisfiable iff $f \in L(s)$ for some node s in the final, pruned tableau, and when f is satisfiable a model for f is contained in the tableau. Thus we have established

Proposition 2 *The above algorithm decides the satisfiability of its input* RTCTL *formula f correctly and in time* $2^{O(|f|)}$. □

Thus, we have a deterministic decision procedure for RTCTL whose complexity is at most exponential in the length of f. Since the problem of determining the satisfiability of CTL formulae is deterministic exponential time complete ((Emerson and Halpern 1982)), and since RTCTL subsumes CTL, our algorithm is essentially optimal:

Theorem 3 *The satisfiability problem for RTCTL is deterministic single exponential time complete.*

Note that the techniques in (Emerson and Clarke 1982) and (Emerson and Halpern 1982) to construct the initial tableau "bottom-up" are applicable to RTCTL as well. Thus the exponential blow-up in $|f|$ need be incurred only in the worst case, rather than in the average case as would be done by the above naive construction of the initial tableau.

5 Other Quantitative Modalities and Temporal Logics

In this section, we briefly consider two other quantitative temporal modalities: $U^{\geq k}$ and $U^{=k}$. Intuitively, $A(p\ U^{\geq k} q)$ says that q is true after k or more time instants along each fullpath and p is true till then. Similarly, $A(p\ U^{=k} q)$ states that q is true exactly at the kth time instant along all fullpaths and p is true at each of the preceding $k-1$ time instants. More formally, we define the logic CRTCTL (Complete RTCTL) to comprise the formulae generated by the rules S1–S4 in Section 2 together with the rules:

S5. If p, q are formulae and k is any natural number, then so are $A(p\ U^{\geq k}\ q)$ and $E(p\ U^{\geq k}\ q)$, and

S6. If p, q are formulae and k is any natural number, then so are $A(p\ U^{=k}\ q)$ and $E(p\ U^{=k}\ q)$.

We also define two sublogics of CRTCTL: RTCTL$^{\geq}$, whose formulae are obtained by using Rules S1–S3 and S5, and RTCTL$^{=}$, whose formulae are generated by Rules S1–S3 and S6.

The semantics of the new quantitative modalities are given by:

S5. $s_0 \models A(p\ U^{\geq k}\ q)$ iff for all fullpaths $s_0, s_1, s_2, \ldots$ in M, $\exists i, i \geq k$, such that $s_i \models q$ and $\forall j, 0 \leq j < i, s_j \models p$
$s_0 \models E(p\ U^{\leq k}\ q)$ iff for some fullpath $s_0, s_1, s_2, \ldots$ in M, $\exists i, i \geq k$, such that $s_i \models q$ and $\forall j, 0 \leq j < i, s_j \models p$

S6. $s_0 \models A(p\ U^{=k}\ q)$ iff for all fullpaths $s_0, s_1, s_2, \ldots$ in M, $s_k \models q$ and $\forall j, 0 \leq j < k, s_j \models p$
$s_0 \models E(p\ U^{=k}\ q)$ iff for some fullpath $s_0, s_1, s_2, \ldots$ in M, $s_k \models q$ and $\forall j, 0 \leq j < k, s_j \models p$

Other abbreviations of these modalities can be defined as in Section 2. It is worth noting that $AF^{=k}p \equiv AG^{=k}p$ is a validity. So is $A(p\,U^{=k}\,q) \Rightarrow A(p\,U^{\leq k}\,q) \wedge A(p\,U^{\geq k}\,q)$, but the same formula with the implication reversed is not valid. We could also define modalities such as $U^{<k}$ and $U^{>k}$, but as we are dealing with discrete time, these are easily expressed in terms of $U^{\leq k}$ and $U^{\geq k}$ respectively.

¿From the semantics above, it is easy to verify that the following formulae are valid. First, for each $k \geq 1$: (*i*) $A(p\,U^{\geq k}\,q) \equiv p \wedge AX\,A(p\,U^{\geq (k-1)}\,q)$; (*ii*) $E(p\,U^{\geq k}\,q) \equiv p \wedge EX\,E(p\,U^{\geq (k-1)}\,q)$; (*iii*) $A(p\,U^{=k}\,q) \equiv p \wedge AX\,A(p\,U^{=(k-1)}\,q)$; and (*iv*) $E(p\,U^{=k}\,q) \equiv p \wedge EX\,E(p\,U^{=(k-1)}\,q)$. Secondly, for $k = 0$: (*v*) $A(p\,U^{\geq 0}\,q) \equiv A(p\,U\,q)$; (*vi*) $E(p\,U^{\geq 0}\,q) \equiv E(p\,U\,q)$; and (*vii*) $A(p\,U^{=k}\,q) \equiv E(p\,U^{=k}\,q) \equiv q$. These formulae may be regarded as the analogues of the fixpoint characterizations of the CTL modalities AU and EU ((Emerson and Halpern 1982)).

We conclude this section with a discussion of results concerning the logics CRTCTL, RTCTL, RTCTL$^{\geq}$, and RTCTL$^{=}$.

Regarding complexity, we can show that there is a polynomial time model checking algorithm for each of the quantitative logics.

Theorem 4 *The model checking problem for the logics RTCTL$^{=}$, RTCTL$^{\geq}$ and CRTCTL can be decided in polynomial time; viz. in time $O(|p||S|^3)$ for input formula p and structure $M = (S, R, L)$.*

Proof: We are assuming that all the constants in the formula are given in binary representation. We show that we can model check for the new modalities given by the rules S5 and S6 within the required time complexity. For each of the subformulas g of the form $A(q\,U^{\geq k}\,r)$ or $A(q\,U^{=k}\,r)$ or $E(q\,U^{\geq k}\,r)$ or $E(q\,U^{=k}\,r)$ we provide a labelling procedure at the end of whose execution, for each state s, the subformula g is in the set $L(s)$ iff s satisfies the subformula. We show that the complexity of executing these labelling procedures is $O(\lceil \log k \rceil |S|^3)$. Since the length of the binary representation of k is $\lceil \log k \rceil$ it will automatically follow that the complexity of the model checking algorithm is $O(|p||S|^3)$. We assume that by the time the labelling procedures for the above subformulas are invoked we have have already executed the labelling procedures for the subformulas q and r.

To label states with subformulas of the form $A(q\,U^{\geq k}\,r)$, we use the identity $A(q\,U^{\geq k}\,r) \equiv A(q\,U^{=k}\,A(q\,U\,r))$ and use the labelling procedures for the subformulas $A(q\,U^{=k}\,r)$ and $A(q\,U\,r)$. To label states with subformulas of the form $E(q\,U^{\geq k}\,r)$ we use the identity $E(q\,U^{\geq k}\,r) \equiv E(q\,U^{=k}\,E(q\,U\,r))$ and use the labelling procedures for the subformulas $E(q\,U^{=k}\,r)$ and $E(q\,U\,r)$.

Now, we describe the labelling procedures for subformulas of the form $A(q\,U^{=k}\,r)$ and of the form $E(q\,U^{=k}\,r)$ that have complexity $O(\lceil \log k \rceil |S|^3)$. If $k = 0$ then s satisfies $A(pU^{=k}q)$ iff s satisfies $E(pU^{=k}q)$ iff s satisfies q. If $k > 0$ then let $l = k - 1$. If $l = 0$ then s satisfies $A(q\,U^{=k}\,r)$ (respectively, satisfies $E(q\,U^{=k}\,r)$) iff s satisfies q and for all (respectively, for some) R-successors s' of s, it is the case that s' satisfies r. In these cases, it should easy to see that we can correctly label the states with the above subformulas in the required time. So, we assume that $l > 0$. Define a new binary relation $T = \{(s, s') : q \in L(s) \cap L(s')\}$. The relation T can be computed in time proportional to $|M|$. For any state $s \in S$, s satisfies $A(q\,U^{=k}\,r)$ (respectively, satisfies $E(q\,U^{=k}\,r)$) iff for all (respectively, for some) states s', s'' such that $(s, s') \in T^l$ and $(s', s'') \in R$, it is the case that s'' satisfies r. Now, we show that the relation T^l can be computed in time $O(\lceil \log k \rceil |S|^3)$. From this it should be easy to see that the we can correctly label states in the required time. We use "iterative squaring" to first compute the relations $T, T^2, T^4, ..., T^{2^m}$ where $m = \lfloor \log k \rfloor$. Note that each of the relation in the above list can be computed by squaring the previous relation and this can be accomplished in time $O(|S|^3)$ using boolean matrix multiplication algorithm. As a consequence all the above relations can be computed in time $O(|S|^3 \lceil \log k \rceil)$. Now the relation T^l can be computed by composing together relations for the appropriate powers of the binary expansion of l. For example, if $l = 5$ then the its binary expansion is 101 and T^l can be computed by composing T^4 and T. In general this involves at most $\lceil \log k \rceil$ additional compositions. and can be accomplished in time $O(|S|^3 \lceil \log k \rceil)$.

$\square$

A decision procedure for the satisfiability problem for RTCTL$^{\geq}$ can be obtained in a manner analogous to that of RTCTL. The validities such as $A(q\,U^{\geq k}\,r) \equiv q \wedge AX\,A(q\,U^{\geq k-1}\,r)$ for $k > 0$ and $A(q\,U^{\geq 0}\,r) \equiv A(q\,U\,r)$ suggest that we define the closure of formula $A(q\,U^{\geq k}\,r)$ to include all subformulas of the form $A(q\,U^{\geq i}\,r)$ for all i such that

$0 \leq i \leq k$. However, if the subformula $A(q\,U^{\geq k'}\,r)$ is in a maximal consistent set of subformulae then we must also have in the set $A(q\,U^{\geq k''}\,r)$ for all smaller k''. Thus, what is significant is the largest k' such that $A(q\,U^{\geq k'}\,r)$ is in the subset. This means that we have only $k+2 = 2^{O(|k|)}$ maximal subsets generated on account of $A(q\,U^{\geq k}\,r)$ and that the tableau is only of single exponential in size. Hence, the complexity works out to be deterministic single exponential time, with the rest of the details similar to those for RTCTL. A corresponding lower bound also follows since RTCTL subsumes CTL which is exponential time-hard (cf. (Emerson and Halpern 1982)). Thus, we have established the following theorem.

Theorem 5 *The satisfiability problem for RTCTL$^{\geq}$ is deterministic single exponential time complete.*

Contrary to the above theorem we see that using precise equality in the quantified modalities costs another exponential as in the case of RTCTL$^{=}$.

Theorem 6 *The satisfiability problem for RTCTL$^{=}$ is double exponential time complete.*

Proof:

The upper bound follows by observing that we can translate a formula in RTCTL$^{=}$ into an equivalent CTL formula whose length is exponential in the length of the original formula, and then applying the exponential time CTL decision procedure.

For the lower bound we argue that every language accepted by an exponential space bounded alternating Turing Machine (ATM) is polynomial time reducible to the satisfiability problem for RTCTL$^{=}$. The lower bound then follows from the results of (Chandra, Kozen, and Stockmeyer 1981) where it is shown that the set of languages that can be recognized in deterministic double exponential time is exactly the set of languages recognized by ATMs using exponential space.

Let $A = (Q, \Sigma, \delta_L, \delta_R, q_0)$ be an ATM that uses space bounded by 2^{cn}, for some constant c, on each input x of length n. The set of states Q is partitioned into the sets Q_{EXIST} of existential states, Q_{UNIV} of universal states, Q_{ACCEPT} of accepting states and Q_{REJECT} of rejecting states. Without loss of generality, we assume that for each universal or existential state q and each symbol $a \in \Sigma$, the finite state control has exactly two moves $\delta_L(q,a)$ and $\delta_R(q,a)$. Each such move is a triple (r, b, D) where r is the next state, symbol b will be the new contents of the current tape cell, and D is either LEFT, RIGHT or NULL indicating that the tape head moves left or right, or stays in the current position, respectively. For each state q which is an accepting or rejecting state there is a single idle transition to the same state leaving the tape contents and the head position unaltered.

A configuration is a sequence of symbols from the set $\Sigma \cup (\Sigma \times Q)$ that contains exactly one symbol from the set $\Sigma \times Q$ and is of length 2^{cn}. If the i^{th} symbol is $a \in \Sigma$ then it indicates the contents of the i^{th} cell to be a; if it is $(a, q) \in \Sigma \times Q$ then it indicates that the head is scanning the i^{th} symbol which contains the value a and the current state is q. We say that a configuration is existential, universal, accepting or rejecting according to its state. We say that a configuration is a successor of another configuration if it is a new configuration that results after the ATM made one move. Note that a universal or existential configuration has exactly two successors and the accepting or rejecting configurations have exactly one successor.

The computation of an ATM A on input x of length n can be represented by a finite tree T whose nodes are configurations of A. Each universal or existential configuration has two successors, and "L-successor" and an "R-successor"¶, while all the accepting or rejecting configurations are terminal nodes in T‖. We label the nodes of the tree with a special proposition AC as follows. Each accepting configuration is labelled with AC; a universal (respectively, existential) configuration is labelled with AC iff both (respectively, one) of its successors are labelled with AC. The ATM A accepts x iff its root is labelled with AC. We could give an $RTCTL^{=}$ formula which defines

¶The L and R "directions" in the tree T should not be confused with the directions LEFT, RIGHT, and NULL that the ATM can move along its itape; they are unrelated.

‖More precisely, we could say that each accepting or rejecting configuration has a single successor that is an identical copy of itself because the ATM "idles" upon reaching an accepting or rejecting configuration.

the tree T, but such a formula will be exponential in length. Instead of directly working with T, we consider another tree T' which is associated with T. Each node (i.e., configuration) of T is represented in T' by a path of length 2^{cn}. We describe an RTCTL$^=$ formula which defines the tree T'.

The root of T' is labelled with a proposition P_{BEGIN}. For each $k \geq 0$, each node at a depth $k2^{cn}$ is labelled with P_{BEGIN}. These are the only nodes where P_{BEGIN} holds. We call such nodes P_{BEGIN} nodes. A proposition P_{END} holds at the parent of each P_{BEGIN} node. We can easily give a formula of length polynomial in n that asserts these properties. For example, the formula $AG(P_{BEGIN} \Rightarrow AX(A(\neg P_{BEGIN} U^{=k} P_{BEGIN})))$, where $k = 2^{cn} - 1$, asserts that the distance between two consecutive P_{BEGIN} nodes is 2^{cn}.

Every path from a P_{BEGIN} node to the next P_{END} node defines a configuration. Corresponding to each $a \in \Sigma$ and $q \in Q$ we have propositions denoted simply as a and q, respectively. We can easily give a formula that asserts that at each point in a configuration exactly one proposition in Σ holds and in every configuration there is exactly one position where some proposition in Q holds and all propositions in Q are mutually exclusive. If at a particular node the propositions q and a are true then it denotes the symbol (a, q) in the configuration. If the proposition a is true and none of the propositions from Q is true then this denotes that the symbol in the corresponding position of the configuration is simply a.

Let t be a P_{END} node. Then every successor node u of t is a P_{BEGIN} node and has the property that either a special proposition L holds on all paths starting from u up to the next P_{END} node or a special proposition R holds on all paths starting from u up to the next P_{END} node. Moreover, there exists such an L-successor u' of t and such an R-successor u'' of t. Each path starting from such an L-successor (respectively, an R-successor) node until the next P_{END} node represents the L-successor (respectively, a R-successor) configuration of the configuration that ended with the node t. This property can be captured by requiring each P_{END} node to satisfy

$$AX(L \vee R) \wedge AX(L \Rightarrow A(L\, U\, (L \wedge P_{END}))) \wedge AX(R \Rightarrow A(R\, U\, (R \wedge P_{END}))) \wedge EX\, L \wedge EX\, R$$

Note that there may be more than one L-successor (R-successor) path. But our formulas will ensure that all such L-successor (R-sucessor, respectively) paths are labelled identically so as to represent the same configuration. It suffices to stipulate for each $a \in \Sigma$ that $AG(EX(a \wedge L) \Rightarrow AX(L \Rightarrow a)) \wedge AG(EX(a \wedge R) \Rightarrow AX(R \Rightarrow a))$, and similarly that for each $q \in Q$, $AG(EX(q \wedge L) \Rightarrow AX(L \Rightarrow q)) \wedge AG(EX(q \wedge R) \Rightarrow AX(R \Rightarrow q))$.

Most importantly, we can also write a formula that relates the tape contents in one cell to the tape contents in the same cell in the next configuration. Since such nodes are separated by a distance of 2^{cn}, we can give such a formula of length polynomial in n. Using this approach we can express the property that each successive configuration represented by a path segment is obtained by a move of A and for every such configuration all its successor configurations occur on different paths. For example, if the propositions $q \in Q$ and $a \in \Sigma$ hold at a partcular node and q is a universal state and $\delta_L(q, a) = (q', a', LEFT)$, then the formula $(q \wedge a) \Rightarrow EF^{=(2^{cn}-1)}(L \wedge q' \wedge EX\, a')$ asserts that the tape symbols and the head position are appropriately changed in the left successor configuration. Similarly if none of the propositions in Q holds at a node then the contents of the corresponding tape symbols in the successor configuration will be identical. This can also be asserted using the formula $(a \wedge \neg state) \Rightarrow EF^{=2^{cn}}(L \wedge a)$ where $state$ is the propositonal formula $\vee\{q : q \in Q\}$.

Finally, to capture proper labelling of the acceptance condition, we can obtain a formula which asserts that the proposition AC holds at the beginning of a universal configuration (respectively, an existential configuration) iff it holds at the beginning of all successor configurations (respectively, some successor configuration) and AC holds at the beginning of accepting configurations and does not hold at the beginning of rejecting configurations. We can also assert that AC holds at the root of the tree. It is fairly straightforward to express all such properties using formulae of length polynomial in n, viz. :

$$\begin{aligned}
&AC \wedge \\
AG((P_{BEGIN} \wedge universal) \Rightarrow (AC \equiv AF^{=2^{cn}}(P_{BEGIN} \wedge AC)])) &\wedge \\
AG((P_{BEGIN} \wedge existential) \Rightarrow (AC \equiv EF^{=2^{cn}}(P_{BEGIN} \wedge AC))) &\wedge \\
AG((P_{BEGIN} \wedge accept) \Rightarrow AC) &\wedge \\
AG((P_{BEGIN} \wedge reject) \Rightarrow \neg AC)&
\end{aligned}$$

where *universal* denotes the formula $A(\neg P_{END}\ U\ \vee\{q : q \in Q_{UNIV}\})$, indicating the state of the configuration is universal, etc. We can also express that the configuration starting with the root of the tree is the initial configuration. Let f be the conjunction of all the above formulae. Clearly, the length of f is polynomial in n and from our construction f is satisfiable iff A accepts the input x. □

Regarding expressive power, we have the following.

Theorem 7 *Each of the quantitative logics RTCTL,RTCTL$^=$,RTCTL$^\geq$ and CRTCTL have equal expressive power coinciding with that of the qualitative logic CTL.* □

Proof: Each of these logics syntactically subsume CTL, and hence are at least as expressive as CTL. Each logic is at most expressive as CTL because each quantitative modalities can be expressed in CTL using repeated applications of the above "fixpoint" characterizations. □

However, the quantitative logics can be exponentially more succinct than CTL because the translation of the bounded modalities into iterated nexttimes amounts to converting the time bound constants from binary to unary. What is interesting is that, despite their exponential succinctness advantage, the quantitative logics can still be model checked in polynomial time. Similarly, testing satisfiability for RTCTL and RTCTL$^\geq$ costs single exponential time which is the same as for CTL. It is only in the case of RTCTL$^=$ and CRTCTL that the exponential succinctness blows the complexity up to double exponential.

6 Conclusion and Related Work

In summary, we have shown how to extend the qualitative logic CTL to the quantitative logic RTCTL. RTCTL permits modalities such as $AF^{\leq k}q$ asserting that q is inevitable within k time units. RTCTL is suitable for time-bounded reasoning about hard real-time computing systems.

We show that RTCTL model checking can be done in time linear in both the formula size and structure size. We believe that our model checking algorithm may be particularly well-suited to reasoning about embedded real-time systems where multiprogramming of multiple software tasks in a uniprocessor hardware environment makes our interleaving semantics quite realistic. In addition, we show that RTCTL satisfiability is decidable in deterministic single exponential time.

We also consider the related logics RTCTL$^\geq$, which permits modalities such as $AF^{\geq k}q$, RTCTL$^=$, which permits modalities such as $AF^{=k}q$, and CRTCTL which permits modalites with bounds specified in terms of $=$, $\leq$, or $\geq$. Model checking for each of these logics can still be done in polynomial time. Satisfiability for RTCTL$^\geq$ can also be decided in single exponential time. Somewhat surprisingly, however, inclusion of the precise equality operator in RTCTL$^=$ and CRTCTL makes their decision problem complete for deterministic *double* exponential time.

Among related work we mention the following. The idea of superscripting or subscripting a linear time temporal operator to get a time bounded modality such as $F^{\leq k}q$ can be traced back to "metric" tense logic studied by philosophers (cf. (van Benthem 1983), (Burgess 1984), (Prior 1957), (Prior 1967)). The application of this natural idea to the specification of real-time systems was explored by Koymans et al ((Koymans, Vytopil, and de Roever 1983) and (Koymans 1990)), giving the specification of a number of example systems using a linear time metric tense logic. Mechanical reasoning is not considered in these papers, however.

Ostroff (Ostroff 1990) considers model checking in the framework of linear time for real-time systems, but not satisfiability testing. In comparing his approach to model checking and ours, we note interesting differences in both the models and the logics. Ostroff uses timed transition systems, where time is measured by the number of clock tick transitions interleaved with ordinary computation transitions, that are compiled into a state reachability graph. In our approach, as in the original papers on model checking of (Clarke and Emerson 1981) and (Clarke, Emerson, and Sistla 1983), we start with a state reachability graph model. To incorporate real-time, our model is extended so that each transition costs a certain amount of time, nominally, unit cost (although nonunit costs can also be handled; cf. Remark 2 following Theorem 1). Ostroff's logic is a partially interpreted first order linear time temporal logic with an explicit clock variable. Our logic, in contrast, is a propostional branching time temporal logic with time bounds incorporated directly into the temporal modalities. When interpreted over finite reachability graphs, the fragment of Ostroff's logic for which a model checking algorithm is given is analogous to a fragment of our logic where path quantifiers are restricted to only be universal. For example, Ostroff's specification $(p \wedge t = T) \Rightarrow \Diamond(q \wedge t \leq T + 5)$ corresponds (roughly) to our formula $p \Rightarrow AF^{\leq 5}q$. (The correspondence is not an exact equivalence because of the different way time is measured in each framework.) Ostroff does not provide model checking algorithms for compound formulae or formulae analogous to $p \Rightarrow EF^{\leq 5}q$, which involve existential path quantification. Indeed, the linear time framework Ostroff has adopted does not permit even the specification of such properties. Such properties involving existential path quantification are useful in specifying lower bounds on nondeterminism and concurrency as well as in ensuring that the logic is closed under semantic negation (cf. (Emerson and Halpern 1983)). We also remark that the satisfiability problem for Ostroff's logic can be shown to be (highly) undecidable (Σ_1^1-hard).

Our work here (cf. (Emerson, Mok, Sistla, and Srinivasan 1989)) appears to be the first to propose the use of time bounded modalities in the branching time framework, and to study the complexity of the the related decision problems (including both model checking and satisfiability testing) with an eye toward applications to real-time systems. Our work derives historically from the work of (Clarke and Emerson 1981). In particular, (Clarke and Emerson 1981) gave a model checking algorithm for (ordinary, qualitative) CTL which implicitly calculated the bounds on the fulfillment of eventualities. However, these bounds were not explicitly stated in the logic or explicitly calculated by the algorithm, and there was no consideration of possible applicability to real-time systems. Moreover, the complexity of the algorithm was quadratic rather than linear. In (Clarke, Emerson, and Sistla 1983) the complexity was improved to be linear. But that algorithm worked by an entirely different method which no longer implicitly calculated the bounds for fulfillment of eventualities. The algorithm presented here runs in linear time and explicitly calculates the eventuality bounds.

Our logic, RTCTL, is a branching time, point-based temporal logic. Modalities such as $AF^{\leq k}q$, however, induce bounded intervals of the form $[0{:}k]$ along computation paths. It is worth noting that a number of (linear time) interval-based logics specialized to facilitate specification of real-time systems have been proposed (cf. (Pnueli and Harel 1988), (Melliar-Smith 1987), (Narayana and Aaby 1988), (Alur, Feder and Henzinger 1991)). Interestingly, the metric interval temporal logic (MITL) of (Alur, Feder and Henzinger 1991) suffers a blowup in the complexity of testing satisfiability when "singular" intervals $[k{:}k]$, i.e., precise equalities, are allowed that is analogous to but far more severe than that encountered in going from RTCTL to RTCTL$^{=}$: MITL becomes (highly) undecidable. The general question of comparing the appropriateness of point-based versus interval-based logics for reasoning about real-time systems is an interesting one that awaits further investigation.

Subsequent work on branching time logics includes (Lewis 1990) which describes a model checking algorithm and its implementation for a CTL-like logic with interval bounded modalities, but does not analyze its complexity or consider testing satisfiability. It permits modalities such as $AF^{[10:25]}q$ meaning that "along every path, after at least 10 time units but within 25 time units, q must occur". All modalities of the (Lewis 1990) logic are succinctly expressible in our logic CRTCTL, and conversely. For example, $AF^{[10:25]}q \equiv AF^{=10}AF^{\leq 15}q$. It follows using our results of Section 5 that this logic's model checking problem is in polynomial time and its satisfiability problem is double exponential time complete.

For our logic RTCTL the underlying semantics of time is discrete, and most of the work discussed above has centered around models where time is discrete. It is also possible to consider real time systems under the assumption that time is dense (or, intuitively, "continuous"). A good deal of more recent work focussing on dense time has been done by Alur [Al91] and Henzinger [He91] and others (cf. (Alur, Courcoubetis, and Dill 1990), (Alur and Dill 1990), (Alur, Feder and Henzinger 1991), (Alur and Henzinger 1989), (Alur and Henzinger 1990)).

One interesting recent effort concerns reasoning about dense time systems specified in TCTL (Timed CTL), proposed in (Alur, Courcoubetis, and Dill 1990). TCTL has almost the same syntax as RTCTL but is interpreted over branching dense time structures. Because of the denseness of time, satisifiability testing for TCTL is highly undecidably (Σ_1^1-hard). Model checking is PSPACE-complete, being polynomial in the size of the state graph, but exponential in the size of the timing constraints. This remains true when the model checking framework is restricted to discrete time. The exponential complexity for TCTL model checking should be contrasted with our polynomial (in fact, linear) RTCTL model checking algorithm. In particular, contrary to the remark in (Alur, Courcoubetis, and Dill 1990), our RTCTL discrete time model checking algorithm does not suffer a similar exponential blowup in the size of the time bounds. An alternative formulation of the TCTL logic is described in (Alur 1991) with syntax analogous to "half-order" modal logic (cf. (Henzinger 1990)).

Finally, the work of (Hansson and Jonsson 1989) and (Hansson 1991) should be mentioned. It involves another real-time logic called PCTL (Probablistic real-time CTL) intended for "soft" deadlines. A typical PCTL expressible property is "with at least 50% probability p will hold within 20 time units." This is written in an essentially linear time syntax as $F^{\leq 20}_{\geq 0.50} p$. A polynomial time model checking algorithm is given and a number of examples are provided.

Other work in the fast growing area of formal approaches to real-time systems including real-time logics, specification languages, and proof methodologies can be found in of real-time logics can be found in (Jahanian and Mok 1986), (Jahanian and Mok 1987), (Gerth and Boucher 1987), (Jahanian and Mok 1988), (Hooman 1991), (Yodaiken and Ramaritham 1990) (Ostroff 1990b), (Ostroff 1991), (Alur 1991), (Henzinger 1991), and (de Roever 1991).

References

[Ab80] Abrahamson, K. 1980. Decidability and Expressiveness of Logics of Processes, Ph.D. Thesis, Univ. of Washington.

[Al91] Alur, R. 1991. Techniques for Automatic Verification of Real-Time Systems, PhD Thesis, Computer Science Department, Stanford University, August, Technical Report STAN-CS-91-1378.

[ACD90] Alur, R., Courcoubetis, C., and Dill, D. 1990. Model-checking for Real-Time Systems, Proc. of the Fifth IEEE Symp. on Logic in Computer Science (LICS), pp. 414-425.

[AD90] Alur, R., and Dill, D. 1990. Automata for Modeling Real-Time Systems, in 17th Inter. Conf. on Automata, Languages, and Programming (ICALP90), M. Paterson, ed., Springer-Verlag LNCS no. 443, pp. 322-335.

[AFH91] Alur, R., Feder, T., and Henzinger, T. 1991. The Benefits of Relaxing Punctuality, Proc. 10th Ann. ACM Symp on Principles of Distributed Computing (PODC), pp. 139-152.

[AH89] Alur, R., and Henzinger, T. 1989. A Really Temporal Logic, Proc. of the 30th IEEE Symp. on Found. of Comp. Sci. (FOCS), pp. 164-169.

[AH90] Alur, R., and Henzinger, T. 1990. Real-Time Logics: Complexity and Expressiveness, Proc. of 5th Ann. Symp. on Logic in Comp. Sci. (LICS), pp. 390-401.

[Br86] Browne, M.C. 1986. An Improved Algorithm for the Automatic Verification of Finite State Systems Using Temporal Logic, *Proc. Symp. on Logic in Computer Science*, Cambridge, pp. 260–266.

[Bu84] Burgess, J. 1984. Basic Tense Logic, in *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner, eds., D. Reidel Pub. Co,

[CKS81] Chandra, A., Kozen, D., and Stockmeyer, L. 1981. Alternation, JACM, vol. 28, no. 1, pp. 114-133.

[CE81] Clarke, E.M. and Emerson, E. A. 1981. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic, *Proc. of the Workshop on Logics of Programs*, Yorktown Heights, D. Kozen, editor, LNCS#131, Springer–Verlag, pp. 52–71.

[CES83] Clarke, E.M., Emerson, E. A. , and Sistla, A. P. 1983. Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications, *Proc. 10th Annual ACM Symp. on Principles of Programming Languages*, Austin, pp. 117–126; *journal version appeared in ACM Transactions on Programming Languages and Systems*, **vol.** 8, no. 2, pp. 244–263, 1986.

[CG87] Clarke, E., and Grumberg, O. 1987. Research on Automatic Verification of Finite State Concurrent Systems, Ann. Rev. Comp. Sci., vol. 2, pp. 269-290.

[CBBG87] Clarke E. M., Bose, S., Browne, M., and Grumberg, O. 1987. The Design and Verification of Finite State Hardware Controllers, Technical Report CMU-CS-87-145, Carnegie-Mellon Univ.

[deR76] de Roever, W.P. 1976. *Recursive Program Schemes: Semantics and Proof Theory*, Mathematical Centre Tracts 70, Mathematisch Centrum, Amsterdam.

[deR91] de Roever, W.-P. 1991., editor, Real-Time: Theory in Practice, Springer-Verlag LNCS, to appear.

[Di76] Dijkstra, E.W. 1976. *A Discipline of Programming*, Prentice–Hall.

[Em90] Emerson, E.A. 1990. Temporal and Modal Logic, in *Handbook of Theoretical Computer Science, vol. B.*, J. van Leeuwen, editor, North–Holland, pp. 995-1072

[EC80] Emerson, E.A. and Clarke, E. M., 1980. Characterizing Correctness Properties of Parallel Programs Using Fixpoints, *Proc. 7th Annual International Colloquium on Automata, Languages and Programming*, LNCS no. 85, Springer–Verlag, pp. 169–181.

[EC82] Emerson, E.A., and Clarke, E. M., 1982. Using Branching Time Logic to Synthesize Synchronization Skeletons, *Science of Computer Programming*, vol. 2, pp. 241–266.

[EH82] Emerson, E.A., and Halpern, J. Y., 1982. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time, *Proc. of the 14th Annual ACM Symp. on Theory of Computing*, San Francisco, pp. 169–180; *also appeared in Journal of Computer and System Sciences*, **vol** 30, no. 1, pp. 1–24, 1985.

[EH83] Emerson, E. A., and Halpern, J. Y., 1983. Sometimes and Not Never Revisited: On Branching versus Linear Time Temporal Logic, Proc. 10th. Annual ACM Symp. on Principles of Programming Languages, Austin, pp. 127-140; journal version appeared in Journal of the ACM, vol. 33, no. 1, pp. 151-178.

[EL85] Emerson, E.A., and Lei, C.-L., 1985. Modalities for Model Checking: Branching Time Logic Strikes Back, *Proc. 12th Annual ACM Symp. on Principles of Programming Languages*, New Orleans, pp. 84–96; *also appeared in Science of Computer Programming*, **vol.** 8, pp. 275–306, 1987.

[EL86] Emerson, E. A., and Lei, C.-L., 1986. Efficient Model Checking in Fragments of the Mu-Calculus, IEEE Symp. on Logic in Computer Science, pp. 267-278.

[EL87] Emerson, E.A., and Lei, C.-L. 1987. New Results on Model-Checking in the Propositional Mu-Calculus, presented at the *Colloquium on Temporal Logic and Specification*, Altrincham, England, April 1987.

[EMSS89] Emerson, E. A., Mok, A. K., Sistla, A. P., and Srinivasan, J. 1989. Quantitative Temporal Reasoning, Proceedings of the Workshop on Automatic Verification Methods for Finite State Systems (Participants Version), C-cube, the French National Concurrency Project, June 12-14, 1989.

[FL79] Fischer, M., and Ladner, R. 1979. Propositional Dynamic Logic of Regular Programs, JCSS, vol. 18, no. 2, pp. 194-211.

[GB87] Gerth, R., and Boucher, A. 1987. A Timed Failures Model for Extending Communicating Processes, In Proc. of the 14th Ann. Int. Conf. on Automata, Languages, and Programming, pp. 95-114, Springer LNCS no. 267.

[HJ89] Hansson, H. and Jonsson, B. 1989. A Framework for Reasoning about Time and Reliability, Proc. of 10th Annual IEEE Real Time Systems Symp., pp. 102-111, Santa Monica, Ca., December 5-7, 1989.

[Ha91] Hansson, H. 1991. Time and Probability in Formal Design of Distributed Systems, PhD Dissertation, Uppsala University, Sweden, DoCS91/27, September 1991.

[He90] Henzinger, T. 1990. Half-Order Modal Logic: How to Prove Real-Time Properties, Proc. of the 9th. Ann. ACM Symp. on Princ. of Distr. Comp. (PODC), pp. 281-296.

[He91] Henzinger, T. 1991. The Temporal Specification and Verification of Real Time Systems, PhD Thesis, Computer Science Department, Stanford University, August 1991, Technical Report STAN-CS-91-1380.

[Ho91] Hooman, J. 1991. Specification and Compositional Verification of Real Time Systems, PhD Thesis, Eindhoven University of Technology.

[JM86] Jahanian, F., and Mok, A. K. 1986. Safety Analysis of Timing Properties in Real Time Systems, IEEE Trans. Software Eng., SE-12(9), pp. 890-904.

[JM87] Jahanian, F., and Mok, A. K. 1987. A Graph-Theoretic Approach for Timing Analysis and its Implementation, *IEEE Transactions on Computers*, **vol.** C–36, no. 8, pp. 961–975.

[JS88] Jahanian, F., and Mok, A. K. 1988. A Method for Verifying Properties of Modechart Specifications, Proc. of the Ninth IEEE Real-Time Systems Symposium, pp. 12-21.

[Koy89] Koymans, R. 1989. Specifying Message Passing and Time Critical Systems with Temporal Logic, PhD Thesis, Eindhoven University of Technology.

[Koy90] Koymans, R. 1990. Specifying Real Time Properties with Metric Temporal Logic, *Real Time Systems*, vol. 2, no. 4., pp. 255-299.

[KVD83] Koymans, R., Vytopil, J., and de Roever, W.-P. 1983. Real Time Programming and Asynchronous Message Passing, Proceedings of the Second Annual ACM Symp. on Principles of Distributed Computing (PODC), pp. 187-197.

[Le90] Lewis, H. R. 1990. A Logic of Concrete Time Intervals, Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS), IEEE Press, pp. 380-399.

[LP85] Lichtenstein, O., and Pnueli, A. 1985. Checking That Finite State Concurrent Programs Satisfy Their Linear Specification, *Proc. 12th Annual ACM Symp. on Principles of Programming Languages,* New Orleans, pp. 97–107.

[LPZ85] Lichtenstein, O., Pnueli, A. and Zuck, L. 1985. The Glory of The Past, *Proc. Conf. on Logics of Programs*, Brooklyn, R. Parikh, editor, LNCS#193, Springer–Verlag, pp. 196–218.

[MW84] Manna, Z., and Wolper, P. 1984. Synthesis of Communicating Processes from Temporal Logic Specifications, *ACM Transactions on Programming Languages and Systems*, **vol.** 6, no. 1, pp. 68–93.

[M-S87] Melliar-Smith, P. M. 1987. Extending Interval Logic to Real Time Systems, Temporal Logic in Specification, B. Banieqbal, H. Barringer, A. Pnueli, eds., pp. 224-242, Springer-Verlag LNCS no. 398, April 1987.

[NA88] Narayana, K. T., and Aaby, A. A. 1988. Specification of Real-Time Systems in Real-Time Temporal Interval Logic, Proc. of IEEE Real-Time Systems Symp., pp. 86-95, Dec. 1988.

[Os90] Ostroff, J. 1990. Deciding Properties of Timed Transition Models, IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 2, pp. 170-183, April 1990.

[Os90b] Ostroff, J. 1990b. Temporal Logic of Real-Time Systems, Wiley, London, 1990.

[Os91] Ostroff, J. 1991. Survey of Formal Methods for the Specification and Design of Real-Time Systems, manuscript, Computer Science Department, York University, Ontario, Canada, to appear in "Tutorial on Specification of Time", IEEE Press, forthcoming.

[Pn77] Pnueli, A. 1977. The Temporal Logic of Programs, *18th Annual Symp. on Foundations of Computer Science*, Providence, pp. 46–57.

[PH88] Pnueli, A., and Harel, E. 1988. Application of Temporal Logic to the Specification of Real-Time Systems, in Formal Techniques in Real-Time and Fault Tolerant Systems, M. Joseph (ed.), Springer-Verlag LNCS no. 331.

[PR89] Pnueli, A., and Rosner, R. 1989. On the Synthesis of a Reactive Module, *Proc. 16th Annual ACM Symp. on Principles of Programming Languages,* Austin, pp. 179–190.

[Pr57] Prior, A. 1957. Time and Modality, Oxford University Press.

[Pr67] Prior, A. 1967. Past, Present, and Future, Oxford University Press.

[QS81] Queille, J.P., and Sifakis, J. 1981. Specification and Verification of Concurrent Systems in CESAR, *Proc. of the 5th International Symposium on Programming*, LNCS no. 137, Springer–Verlag, pp. 337–350.

[SC82] Sistla, A.P., and Clarke, E. M. 1985. The Complexity of Propositional Linear Temporal Logics, *Proc. of the 14th Annual ACM Symp. on Theory of Computing*, San Francisco, pp. 159–168, 1982; *also appeared in Journal ACM*, **vol.** 32, no. 3, pp. 733–749.

[vB83] van Benthem, J. 1983. The Logic of Time, D. Reidel Pub. Co.

[YR90] Yodaiken, V., and Ramamritham, K. 1990. Specifying and Verifiying a Real-Time Priority Queue with Modal Algebra, Proc. 11th IEEE Symp on Real-Time Systems, pp. 300-311, Dec. 5-7, 1990.