Reasoning about Extremal Properties of Events

Jatindra Kumar Deka
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati, INDIA 781039
jatin@iitg.ernet.in

Abstract

This paper deals with a branching time temporal query language called Min-max CTL which is similar in syntax to the popular temporal logic, CTL [7]. Min-max CTL can express timing queries on a timed model, whereas CTL is used for untimed systems. Interesting timing queries involving a combination of min and max can be expressed in Min-max CTL. While model checking using most timed temporal logics is PSPACE complete or harder [3, 2], it is shown in [10] that many practical timing queries, where we are interested in the worst case or best case timings, can be answered in polynomial time by querying the system using Min-max CTL. In this paper the syntax of Min-max CTL is extended to increase the expressive power of Min-max CTL.

1 Introduction

Temporal logic model-checking [7] is one of the most popular and well studied paradigms for formal verification of hardware and other concurrent systems [8]. In this approach, each concurrent process is modeled as a finite state non-deterministic transition system. The correctness property which needs to be verified on the given set of concurrent transition systems is specified as 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, 8].

Traditional temporal logics such as *Linear Temporal Logic* (LTL), *Computation Tree Logic* (CTL), and CTL* [7] can be used to reason about the temporal behavior of systems without explicitly quantifying time. These logics are therefore inadequate to quantitatively reason about timing properties of a system. On the other hand timed temporal logics such as *Timed CTL* (TCTL) [3] and *Real Time CTL* (RTCTL) [11] allow us to verify actual timing properties on a timed transition system. However, the verification of timed temporal logics has been found to be much more complex than their untimed counterparts [1]. For example, while LTL model checking is PSPACE complete, TLTL model checking is undecidable [4]. The problem is less severe in the case of branching time timed logics, where TCTL model checking is PSPACE complete [3, 2] (where as CTL model checking is possible in polynomial time). It has been shown in [2] that TCTL model checking is PSPACE complete even in discrete-time models.

While analyzing timed transition systems, we often like to reason about the extremal (best case or worst case) temporal properties of the system. Determining such temporal properties can also be used as a strategy for verification [6]. In [10], we proposed a temporal query language called Min-max CTL which allows the quantification of CTL state and path properties in terms of a cost function over real time. Min-max CTL is capable of expressing queries which involve a combination of extremal (min and max) quantifiers both along paths (or walks) in the timed model, as well as across paths. For example, given the air, rail and road maps of a country represented as graphs with the cities as vertices and links labeled with the duration of travel, we can express queries such as the following:

> *What is the minimum time to travel from city s to city z, when we travel solely by rail or road? What is the minimum total time to travel from city s to city z, when we travel as far as possible by air, and then go by rail or road?*

It is also possible to express complex queries like,

> *Determine the minimum among the worst-case response times for the earliest* request *along all possible computation paths of a client process, where the worst-case response time for a* request *of a client is defined as the maximum delay by which the* grant *is given by the server. Here* request *and* grant *are treated as events.*

Figure 1 shows a sample timed model for the query presented above related to *Client-Server*, where the answer to

Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and
Fourth International Conference on Temporal Logic (TIME-ICTL'03) 1530-1311/03 $17.00 © 2003 IEEE

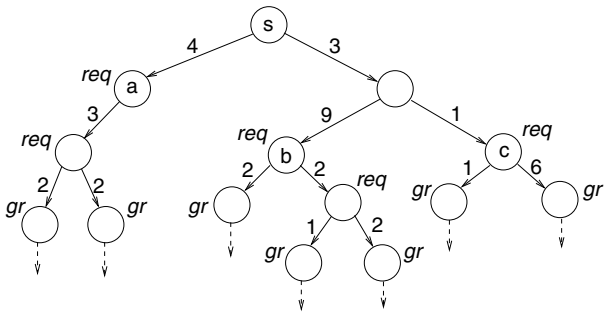

Figure 1. Sample Timed model: Client-Server

this query is 4, since the worst-case response time of the requests at states a, b, and c, are 5, 4, and 6 respectively.

We introduced the Min-max CTL in [10]. We showed in [10] that a category of Min-max CTL (which covers many practical min-max timing queries of interest) can be evaluated in polynomial time though the general problem is DP-hard.

In this paper we describe three extensions of Min-max CTL, which are the main contribution of this paper. The main features of these extensions may be summarized as follows:

- In the basic syntax of Min-max CTL we have not considered the conjunction and disjunction of Min-max CTL formulas, since the semantics of evaluation does not naturally match with the semantics of the CTL restrictions of such formulas. In this paper we address this issue and show that a non-ambiguous semantics can be defined with proper restrictions on the cost function.

- The syntax of Min-max CTL in the basic formulation does not permit a Min-max CTL formula before the until operator. That is, in all formulas of the form $(f_1\ U\ f_2)$, f_1 can only be a CTL formula. In this paper we present an interesting extension which allows us to quantify the values evaluated along the path when f_1 is also a Min-max CTL formula.

- We also present an extension of Min-max CTL which allows the normal CTL until operator within a quantified Min-max CTL formula. For example, we allow formulas of the type $\min E_{g+h}(p\ \ U\ \ (\min E_g(q\ \ U_{\min}\ \ r)))$, which is not expressible in the Min-max CTL introduced in [10].

This paper is organized as follows. In Section 2 we provide a brief introduction of basic Min-max CTL. The basic syntax of Min-max CTL is extended to allow conjunction and disjunction of Min-max CTL formulas in Section 3. Section 4 presents the extension which allows us to quantify the values evaluated along the path when a Min-max CTL formula is used before an until operator. Section 5 presents an extension of Min-max CTL which allows the normal CTL until operator within a quantified Min-max CTL formula. Section 6 provides an outline of the algorithm to evaluate the queries of extended Min-max CTL. Section 7 presents the conclusions of this paper.

2 Min-max CTL

The syntax of Min-max CTL is similar to that of CTL, except for two special types of *until* operators, which are called *min-until* ($U_{\min}$) and *max-until* ($U_{\max}$) respectively, and two special types of existential (E) and universal (A) quantifiers, which (like CTL) must immediately precede each *min-until* and *max-until* path formula. These special quantifiers act as min or max operators across paths. Generally the next-time (X) operator of CTL is excluded in case of timed models.

2.1 Syntax

In this section we describe the model and the formal syntax of Min-max CTL.

Definition 2.1 [Timed Model:]
A timed model is a tuple $J = \langle \mathcal{AP}, S, \mathcal{R}, s_0, \mathcal{L} \rangle$, where:

- *$\mathcal{AP}$ is the set of atomic propositions,*
- *S is the finite set of states,*
- *$\mathcal{R} \subseteq S \times S \times \mathcal{N}$ is a transition relation, where $\mathcal{N}$ denotes the set of positive integers, and $(s_i, s_j, \tau_{ij}) \in \mathcal{R}$ implies that the delay between successive states s_i and s_j is τ_{ij} units of time,*
- *$s_0 \in S$ is the initial state,*
- *$\mathcal{L} : S \rightarrow 2^{\mathcal{AP}}$ is a labeling of states with atomic propositions true in that state.*

The syntax of Min-max CTL is as follows. B denotes boolean formulas over the set of atomic propositions $q \in \mathcal{AP}$, S denotes CTL formulas, and Z denotes Min-max CTL formulas. C and C' are user defined functions.

- $B = false|true|q|\neg q|B \wedge B|B \vee B|\neg B$
- $S = B|S \wedge S|S \vee S|E(S\ U\ S)|A(S\ U\ S)|\neg S$
- $Q = \min | \max$
- $Z = Z \wedge S|QE_{C(g,h)}(SU_QZ)|QE_{C'(g)}(SU_QS)|$ $QA_{C(g,h)}(SU_QZ)|QA_{C'(g)}(SU_QS)$

Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and
Fourth International Conference on Temporal Logic (TIME-ICTL'03) 1530-1311/03 $17.00 © 2003 IEEE

We use the following abbreviations:

$F\ q$: $\quad true\ U\ q$
$F_{\min}\ q$: $\quad true\ U_{\min}\ q$
$F_{\max}\ q$: $\quad true\ U_{\max}\ q$

Throughout this paper, Z-formulas, S-formulas and B-formulas refer to formulas derived out of Z, S and B respectively.

In traditional CTL model checking we seek to determine whether a given CTL formula (of the form of S) is true at a state of the model. In Min-max CTL, we have extended the syntax of CTL by allowing the quantified until operators ($U_{\min}$ and $U_{\max}$) and the min and max path quantifiers, which are defined over the cost functions C or C'. Thereby Min-max CTL also has a semantics of evaluation which returns a numeric value whenever the formula is true at a state. This numeric value quantifies the value of the objective function C (or C') which we seek to optimize, and is the answer to the Min-max CTL query at that state. On the other hand, if the formula is false at a state of the model, then the evaluation of the formula at that state returns null.

A Min-max CTL formula is said to be true at a state when the *CTL restriction* of the Min-max CTL formula is true at that state. This restriction is obtained by removing the functions C, C' and the quantifiers Q and Q' from the Min-max CTL formula. We present a couple of examples to illustrate the CTL restriction of a Min-max CTL formula:

- The CTL restriction of the Min-max CTL formula $\max A_{2g}(F_{\min}\ q)$ is AFq.
- The CTL restriction of the Min-max CTL formula $\min E_{g+h}(p\ U_{\min}\ (\min E_g(q\ U_{\min}\ r)))$ is $E(p\,U\,(E(q\,U\,r)))$.

Min-max CTL evaluation returns a numeric *Min-max value* corresponding to the Min-max CTL formula at states where the CTL restriction of the formula is true. The semantics of Min-max CTL and the algorithm to evaluate the value of Min-max queries are presented in [10]. It is also shown that in general the problem is DP-hard, but there are important special cases which cover most of the practical queries of interest and can be evaluated in polynomial time. This category is termed as monotonic Min-max CTL, where the cost function is monotonic in nature.

We illustrate the semantics of evaluation through an example, the formal definition of semantics is presented in [10].

- Figure 2 shows a sample timed model. Suppose we are interested in determining the earliest occurrence of a state where q is true, starting from state a. We can pose this query as $\min E_g(F_{\min}\ q)$ to be evaluated at state a. Using the $U_{\min}$ operator indicates that we are interested in the earliest occurrence of q along the path. Hence the path formula $F_{\min}\ q$ evaluates to 4 on all paths through state b. This value is the delay to reach state b from state a. We refer to the state b as the *closing state* of these paths. Similarly, for all paths which go through state e and not through state b, the path formula $F_{\min}\ q$ evaluates to 6, and e is the closing state. The delay from the start state of the path to the closing state is called the g-value of the path. The quantifier $\min E_g$ specifies that the path with minimum g-value has to be chosen, which in this case is any path through state b. Hence the Min-max CTL formula $\min E_g(F_{\min}\ q)$ evaluates to 4 at state a.

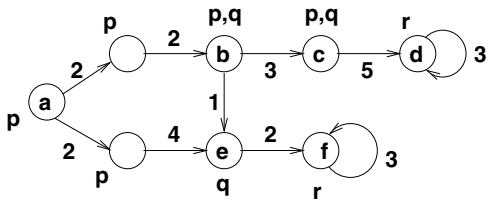

Figure 2. A timed model

- Consider the query $\min E_{2g}(F_{\min}\ q)$. This query differs from the previous one only in the cost function, which is $2g$ instead of g. The best path is still the same, but the value returned at state a is now 8.
- Using the $U_{\max}$ operator, we can determine the latest occurrence of q along a path. For example, in Figure 2, evaluating the formula $\min E_g(F_{\max}\ q)$ at state a yields 5 and the best path is the one through state b followed by state e. State e is the closing state of the path.
- Consider the Min-max CTL formula $\varphi = \min E_{g+h}(p\ U_{\min}\ (\min E_g(q\ U_{\min}\ r)))$. Here we use the cost function $C(g,h) = g + h$. The h-value of a state with respect to φ is the Min-max value computed at that state for the subformula $\psi = \min E_g(q\ U_{\min}\ r)$. Since we use $U_{\min}$, the possible closing states for the path formula $p\,U_{\min}\,\psi$ are b (for all paths through b) and e (for all paths through e which do not go through b). Evaluation of ψ yields $h = 3$ at state b and $h = 2$ at state e. Thus for all paths where b is the closing state, $C(g,h) = g + h = 7$, and for all paths where e is the closing state, $C(g,h) = g + h = 8$. Since the path quantifier is $\min E_{g+h}$, the value returned by evaluating φ at state a is 7.

Thus the numeric values returned by the evaluation of a Min-max CTL formula f at states of the timed model are determined by the values (namely h-values) returned by

evaluating the subformulas of f at the states of the model and the *lengths* (namely g-values) computed by evaluating the special until operator U_Q (which may be $U_{\min}$ or $U_{\max}$) over the paths of the model. The objective function may have two parameters, namely g and h. The g-value is associated with the path cost computed by virtue of the U_Q operator, and the h-value is associated with evaluating the subformula following the U_Q operator. The user defined objective functions C and C' compute the merits of paths as a function of the g-values and h-values, and the min and max path quantifiers specifies the criterion for choosing between values returned by evaluating these paths.

3 Conjunction and Disjunction in Min-max CTL

Min-max CTL evaluation may be extended to the conjunction and disjunction of Min-max CTL subformulas in many ways. In this section we highlight the main issues and present one such extension of Min-max CTL for conjunction and disjunction.

The basic semantics of the proposed extension is to combine the values evaluated by Min-max CTL subformulas, f_1 and f_2 to obtain the value evaluated by $f_1 \vee f_2$ or $f_1 \wedge f_2$. The combined cost function can be defined in many ways, but must consider the cases where the CTL restriction of either f_1 or f_2 is not true at the state, in which case the value evaluated by that subformula will be $null$. We suggest one possible way of choosing such an evaluation semantics which is motivated by the following example.

Example 3.1 Consider the timed model in Figure 3. The states labeled t are *terminal states*, the states labeled c are *control states*, and the states labeled o are *observable states*. The delays are shown on the edges.

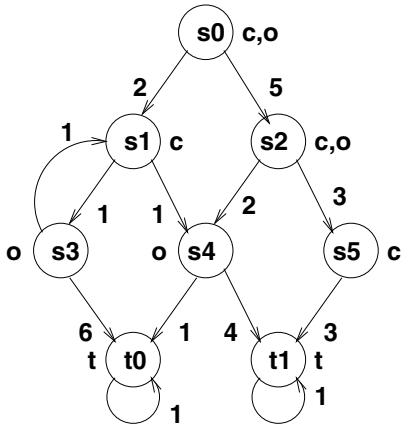

Figure 3. Timed model – 1

Suppose we are interested in determining the minimum delay to reach a terminal state through a path consisting of either control states or observable states. This may be expressed in Min-max CTL as follows:

$$\min E_g((c \vee o)\ U_{\min}\ t)$$

In Figure 3 this query evaluates to 4 at the state s_0, and corresponds to the path s_0, s_1, s_4, t_0. Now suppose we are interested in determining the minimum delay to reach a terminal state through a path consisting of solely control states or solely observable states. This property cannot be expressed in the basic syntax of Min-max CTL in a single query. We can however, extend Min-max CTL to specify this property as follows:

$$\varphi = (\min E_g(c\ U_{\min}\ t)) \bigvee_{\min} (\min E_g(o\ U_{\min}\ t))$$

The operator $\bigvee_{\min}$ computes the minimum of the values evaluated by its two operands. If both of them is $null$, then it returns $null$ since the CTL restriction of φ does not hold. If one of them is $null$, then the CTL restriction of φ holds, and hence the value evaluated by the other operand is returned. In this example, the value returned by $\min E_g(c\ U_{\min}\ t)$ at s_0 is 11, and corresponds to the path s_0, s_2, s_5, t_1. The value returned by $\min E_g(o\ U_{\min}\ t)$ at s_0 is 8, and corresponds to the path s_0, s_2, s_4, t_0. Thereby the value evaluated for φ at s_0 is 8.

Now, consider the Min-max CTL formulas:

$$\begin{aligned} \psi_1 &= \min E_g((c \wedge o)\ U_{\min}\ t) \\ \psi_2 &= \min E_g(true\ U_{\min}\ t) \end{aligned}$$

The CTL restriction of ψ_1 does not hold at s_0, while the CTL restriction of ψ_2 holds at s_0. Since the CTL restriction of $\psi_1 \wedge \psi_2$ does not hold at s_0, the evaluation of the formula $\psi_1 \bigwedge_{\min} \psi_2$ at s_0 will return $null$. On the other hand, the CTL restriction of $\psi_1 \vee \psi_2$ holds at s_0, and the evaluation of $\psi_1 \bigvee_{\min} \psi_2$ will evaluate to 4 (which is the value obtained by evaluating ψ_2 at s_0).

The semantics of evaluation for the operators $\bigvee_{\min}$, $\bigvee_{\max}$, $\bigwedge_{\min}$ and $\bigwedge_{\max}$ are as follows:

$$\begin{aligned} EvalS(f_1 \bigwedge_{\min} f_2, s) &= null \\ &\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \text{ or} \\ &\quad \text{EvalS}(f_2\text{,s}) \text{ is } null \\ &= \min\{EvalS(f_1, s), EvalS(f_2, s)\} \\ &\quad otherwise \\ EvalS(f_1 \bigwedge_{\max} f_2, s) &= null \\ &\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \text{ or} \\ &\quad \text{EvalS}(f_2\text{,s}) \text{ is } null \end{aligned}$$

$$
\begin{aligned}
&= \max\{EvalS(f_1,s), EvalS(f_2,s)\} \\
&\quad otherwise \\
EvalS(f_1 \bigvee_{\min} f_2, s) &= null \\
&\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \text{ and} \\
&\quad \text{EvalS}(f_2\text{,s}) \text{ is } null \\
&= EvalS(f_1,s) \\
&\quad \text{if EvalS}(f_2\text{,s}) \text{ is } null \\
&= EvalS(f_2,s) \\
&\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \\
&= \min\{EvalS(f_1,s), EvalS(f_2,s)\} \\
&\quad otherwise \\
EvalS(f_1 \bigvee_{\max} f_2, s) &= null \\
&\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \text{ and} \\
&\quad \text{EvalS}(f_2\text{,s}) \text{ is } null \\
&= EvalS(f_1,s) \\
&\quad \text{if EvalS}(f_2\text{,s}) \text{ is } null \\
&= EvalS(f_2,s) \\
&\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \\
&= \max\{EvalS(f_1,s), EvalS(f_2,s)\} \\
&\quad otherwise
\end{aligned}
$$

The function $EvalS(f,s)$ is used to denote the value evaluated by the Min-max CTL state formula, f, on a state, s, of the timed model. It is easy to see that this extension to Min-max CTL does not lead to any increase in its evaluation complexity. The Algorithm, that is presented in [10], evaluates the value of a Min-max formula after getting the value of its all subformulas. Because of this evaluating style, the complexity of the algorithm for this extension remains the same. There are several ways in which similar extensions may be done. For example, additive costs structures similar to AND/OR graphs [12] can be modeled using addition of $EvalS(f_1,s)$ and $EvalS(f_2,s)$, instead of using $\min$ or $\max$. We have suggested a $\min$ or $\max$ extension in keeping with the basic style of Min-max CTL.

In the more general case, we may define the semantics of evaluation for the operators $\bigwedge_{C(g_1,g_2)}$ and $\bigvee_{C(g_1,g_2)}$ as follows:

$$
\begin{aligned}
EvalS(f_1 \bigwedge_{C(g_1,g_2)} f_2, s) &= null \\
&\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \text{ or} \\
&\quad \text{EvalS}(f_2\text{,s}) \text{ is } null \\
&= C\{EvalS(f_1,s), EvalS(f_2,s)\} \\
&\quad otherwise \\
EvalS(f_1 \bigvee_{C(g_1,g_2)} f_2, s) &= null \\
&\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \text{ and} \\
&\quad \text{EvalS}(f_2\text{,s}) \text{ is } null \\
&= EvalS(f_1,s) \\
&\quad \text{if EvalS}(f_2\text{,s}) \text{ is } null \\
&= EvalS(f_2,s) \\
&\quad \text{if EvalS}(f_1\text{,s}) \text{ is } null \\
&= C\{EvalS(f_1,s), EvalS(f_2,s)\} \\
&\quad otherwise
\end{aligned}
$$

The cost function C has to be defined properly so that it can evaluate the value.

In the basic formulation of Min-max CTL, we had not allowed the disjunction of a CTL subformula, ψ, with a Min-max CTL subformula, φ. This restriction was imposed to avoid a situation where the CTL restriction of $\psi \vee \varphi$ holds due to only ψ holding at that state. In such a situation evaluation of the Min-max CTL formula $\psi \vee \varphi$ at a state could return $null$ even though its CTL restriction holds at that state. Defining the semantics on the lines shown in this section, we can extend Min-max CTL to allow disjunctions of CTL and Min-max CTL subformulas.

4 Quantifying Min-max CTL over paths

In the syntax of the basic Min-max CTL, formulas of the form $(f_1\ U\ f_2)$ are restricted to cases where f_1 is a CTL formula. In a path where $(f_1\ U\ f_2)$ is true, we may have multiple states where f_1 is true. In this section we show that if we allow f_1 to be a Min-max CTL formula, and quantify (again through min or max) the values evaluated on the states labeled f_1 preceding f_2 on the path, then the expressive power of Min-max CTL can be significantly enhanced.

Example 4.1 Consider the timed model in Figure 4. The states labeled t are *terminal states*, the states labeled c are *control states*, and the states labeled o are *observable states*. The delays are shown on the edges.

In Figure 4 there are two paths from s_0 to terminal states through control states only. These are s_0, s_1, s_3, t_0 and s_0, s_2, s_5, t_1 respectively. Now suppose we are interested in choosing the path which is close to observable states. Let $d(s)$ denote the minimum delay from state s to any observable state. Let $d(\pi)$ denote the maximum distance of path π from observable states, which is defined as $\max\{d(s), s \in \pi\}$. We are interested in the path, π, from s_0 to a terminal state (through control states), for which $d(\pi)$ is minimum, that is, the path for which the maximum distance to observable states is minimized. This query cannot be expressed in the basic version of Min-max CTL. We propose to extend Min-max CTL to express this property as follows:

$$\min E_k\,(\max[c \wedge \min E_g(true\ U_{\min}\ o)]\ U_{\min}\ t)$$

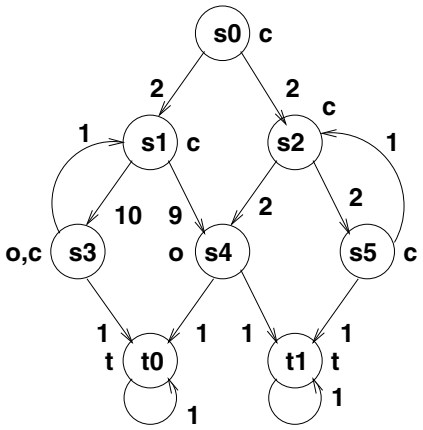

Figure 4. Timed model – 2

Evaluating $\min E_g(true U_{\min} o)$ at the states s_0, s_1 and s_3 returns 4, 9 and 0 respectively. Thereby, on the path s_0, s_1, s_3, t_0, $\max[c \wedge \min E_g(true\ U_{\min}\ o)]$ evaluates to $k = \max\{4, 9, 0\} = 9$. Similarly, on the path s_0, s_2, s_5, t_1, $\max[c \wedge \min E_g(true\ U_{\min}\ o)]$ evaluates to $k = \max\{4, 2, 3\} = 4$. The $\min E_k$ quantifier requires us to choose the path with minimum k. Hence the path s_0, s_2, s_5, t_1 is the desired path having a cost of 4.

In a similar way if we are interested in determining a path from s_0 to a terminal state through control states for which the minimum distance to observable states is minimized, we can write this as:

$$\min E_k(\min[c \wedge \min E_g(true\ U_{\min}\ o)]\ U_{\min}\ t)$$

The k value for the paths s_0, s_1, s_3, t_0 and s_0, s_2, s_5, t_1 are now 0 and 2 respectively. Hence the desired path is s_0, s_1, s_3, t_0 having a cost of 0.

If we wish to determine a path from s_0 to a terminal state through control states for which the minimum distance to observable states is maximized, we may write:

$$\max E_k(\min[c \wedge \min E_g(true\ U_{\min}\ o)]\ U_{\min}\ t)$$

For this query the desired path is s_0, s_2, s_5, t_1 having a cost of 2.

In the extension described in this Section, we introduce path formulas of the form $Q''[f_1]\ U_{Q'}\ f_2$, where Q'' and Q' are min or max. The semantics of evaluating such a formula over a path π which satisfies the CTL restriction of this formula is as follows. Consider the values returned by evaluating f_1 on the states in π preceding the closing state (that is, the state where f_2 is true). We define a quantity k (just as we defined g and h for Min-max CTL evaluation), which denotes the minimum or maximum (depending on Q'') of these values. The cost function is now a function of g, h and k.

4.1 Syntax and Semantics

The formal syntax of this extension of Min-max CTL is as follows. B denotes the set of boolean formulas over the set of atomic propositions $q \in \mathcal{AP}$, S denotes CTL formulas, and Z denotes Min-max CTL formulas. C and C' are user defined functions.

- $B = false \mid true \mid q \mid \neg q \mid B \wedge B \mid B \vee B \mid \neg B$
- $S = B \mid S \wedge S \mid S \vee S \mid E(S\ U\ S) \mid A(S\ U\ S) \mid \neg S$
- $Q = \min \mid \max$
- $Z = Z \wedge S \mid QE_{C(g,h)}(S\ U_Q\ Z) \mid QE_{C'(g)}(S\ U_Q\ S) \mid QA_{C(g,h)}(S\ U_Q\ Z) \mid QA_{C'(g)}(S\ U_Q\ S) \mid QE_{C(g,h,k)}(Q[Z]\ U_Q\ Z) \mid QA_{C(g,h,k)}(Q[Z]\ U_Q\ Z) \mid QE_{C'(g,k)}(Q[Z]\ U_Q\ S) \mid QA_{C'(g,k)}(Q[Z]\ U_Q\ S)$

Each Min-max CTL state formula has an embedded cost function C. In the basic formulation of Min-max CTL, the cost function had at most two parameters, namely g which represented the path cost, and h which represented the cost evaluated at the closing state of the path. In this extension, the cost function C may have (in addition to g and h) another parameter namely k. For a path formula of the form $Q''[f_1]\ U_{Q'}\ f_2$, the parameter k evaluates to the minimum (if Q'' is min) or maximum (if Q'' is max) of the values returned by evaluating f_1 on the states preceding the closing state of a satisfying path.

We use the function, $EvalS(f, s)$ to denote the value evaluated by the Min-max CTL state formula, f, on a state, s, of the timed model. As described earlier, each Min-max CTL state formula has an embedded cost function, C, which is a function of the g-value of a path, the h-value of the closing state of the path and the k-value of the path.

We use the function, $EvalP(f, \pi, C)$ to denote the value evaluated by the Min-max CTL path formula, f, on a path, π, of the timed model, with respect to the cost function, C. Since each *until* in a Min-max CTL formula must be immediately preceded by an A or E, the cost function C is always well defined for each path sub-formula.

The evaluation, $EvalP(f, \pi, C)$, of an extended Min-max CTL path formula, f, on a path, π, with respect to a cost function, C, is defined as follows. Whenever $\pi \not\models f$, $EvalP$ returns $null$. Otherwise, we have the following cases.

Case-1: $f = \min[f_1]\ U_{min}\ f_2$: Since the U_{min} operator has been used, the closing state, t, of π is defined as the earliest state in π such that $t \models f_2$ and for each state w preceding t in π, $w \models f_1$ (and since t is the earliest such state, $w \not\models f_2$). The g-value of π is the total delay from the starting state of π to state t. Since f_1 is a Min-max CTL formula, evaluation of f_1 (using

$EvalS(f_1, w)$) at each node w returns a value. The k-value of the path is the minimum among these values. The state w in π which returns this minimum value is called the *contributing state* of π.

We have two cases under this, namely if f_2 is a CTL formula, or if f_2 is a Min-max CTL formula.

Case-1A: f_2 is CTL: In this case, by definition of Min-max CTL, the function C must be a function of the g-value of π and the k-value of π. $EvalP(f, \pi, C)$ evaluates to the value of $C(g, k)$.

Case-1B: f_2 is Min-max CTL: In this case, the function C is a function of the g-value of π, the k-value of π, and the h-value computed by $EvalS(f_2, t)$, where t is the closing state of π. $EvalP(f, \pi, C)$ evaluates to the value of $C(g, h, k)$.

Case-2: $f = \max[f_1]\, U_{min}\, f_2$: This situation is similar to the Case-1, except that we have a max quantifier before f_1. Therefore, the k-value of the path is the maximum among the values returned by $EvalS(f_1, w)$ among the states w preceding the closing state of the path. The state w in π which returns this maximum value is called the *contributing state* of π.

Again we have two cases depending on whether f_2 is CTL or Min-max CTL.

Case-2A: f_2 is CTL: This is similar to Case-1A, and $EvalP(f, \pi, C)$ evaluates to the value of $C(g, k)$.

Case-2B: f_2 is Min-max CTL: This is similar to Case-1B, and $EvalP(f, \pi, C)$ evaluates to the value of $C(g, h, k)$.

Case-3: $f = min[f_1]\, U_{max}\, f_2$: Since the U_{max} operator has been used, the closing state, t, of π is the last state in π such that $t \models f_2$ and for each state w preceding t in π, $w \models f_1$. Such a last state is well defined except when for each state w of π, $w \models f_1$ and some state t' such that $t' \models f_2$ occurs infinitely often in π (that is, the path describes a cycle through t'). In such cases the g-value of π is taken to be ∞. Otherwise the g-value of π is the total delay from the starting state of π to the closing state t. Since f_1 is a Min-max CTL formula, evaluation of f_1 (using $EvalS(f_1, w)$) at each node w returns a value. The k-value of the path is the minimum among these values. The state w in π which returns this minimum value is called the *contributing state* of π.

Again we have two cases depending on whether f_2 is CTL or Min-max CTL.

Case-3A: f_2 is CTL: This is similar to Case-1A, and $EvalP(f, \pi, C)$ evaluates to the value of $C(g, k)$.

Case-3B: f_2 is Min-max CTL: This is similar to Case-1B, and $EvalP(f, \pi, C)$ evaluates to the value of $C(g, h, k)$.

Case-4: $f = max[f_1]\, U_{max}\, f_2$: This situation is similar to Case-3, except that we have a max quantifier before f_1 instead of a min quantifier. Therefore, the k-value of the path is the maximum among the values returned by $EvalS(f_1, w)$ among the states w preceding the closing state of the path. The state w in π which returns this maximum value is called the *contributing state* of π.

Again we have two cases depending on whether f_2 is CTL or Min-max CTL.

Case-4A: f_2 is CTL: This is similar to Case-1A, and $EvalP(f, \pi, C)$ evaluates to the value of $C(g, k)$.

Case-4B: f_2 is Min-max CTL: This is similar to Case-1B, and $EvalP(f, \pi, C)$ evaluates to the value of $C(g, h, k)$.

The evaluation of a Min-max CTL state formula, f, on a state, s, is as follows:

Case-1: $f = \min E_C(f')$: In this case, $EvalS(f, s)$ evaluates to the minimum among the non-null values returned by $EvalP(f', \pi, C)$ for each path, π, starting at s.

Case-2: $f = \max E_C(f')$: In this case, $EvalS(f, s)$ evaluates to the maximum among the non-null values returned by $EvalP(f', \pi, C)$ for each path, π, starting at s.

Case-3: $f = \min A_C(f')$: In this case, $EvalS(f, s)$ evaluates to the minimum among the values returned by $EvalP(f', \pi, C)$ for each path, π, starting at s. Note that since $s \models f$, $EvalP(f', \pi, C)$ will not return null for any of the paths, π.

Case-4: $f = \max A_C(f')$: In this case, $EvalS(f, s)$ evaluates to the maximum among the values returned by $EvalP(f', \pi, C)$ for each path, π, starting at s. Note that since $s \models f$, $EvalP(f', \pi, C)$ will not return null for any of the paths, π.

5 Un-quantified *until-operator* in Min-max CTL

The syntax of basic Min-max CTL allows evaluation of path formulas having only the quantified until operators,

$U_{\min}$ and $U_{\max}$. In this section we show that allowing the un-quantified CTL until operator U in Min-max CTL evaluation enhances the expressive power of Min-max CTL without any significant compromise on the computational complexity of evaluation. We first illustrate the utility of this extension through an example.

Example 5.1 Consider the timed model shown in Figure 5. The states labeled by *req* represents requesting states (for some resource), and the states labeled by *gr* represents states where the resource is granted. The delays are shown besides the edges. The delay from a request state, s to the earliest grant state, w, following s on a path π will be referred to as the *response time* for s along π.

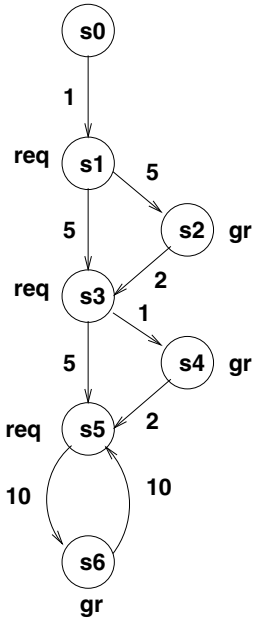

Figure 5. Timed model – 3

Suppose we wish to determine the best response time from the earliest requesting state. This query may be expressed in the basic formulation of Min-max CTL as follows:

$$\min E_h(true\ U_{\min}\ (req \wedge \min E_g(true\ U_{\min}\ gr)))$$

Since the earliest requesting state is s_1, the value evaluated for this query is the best response time from s_1, that is, 5. Similarly, if we wish to determine the best response time from the last requesting state, we may express the query in the basic Min-max CTL as follows:

$$\min E_h(true\ U_{\max}\ (req \wedge \min E_g(true\ U_{\min}\ gr)))$$

Since the last requesting state is always s_5, the value evaluated for this query is the best response time from s_5, that is, 10.

Now suppose we wish to determine the best response time from any requesting state reachable from s_0 (not necessarily the earliest or the latest). The basic formulation of Min-max CTL cannot express this query, since all *until* operators used in evaluation are either $U_{\min}$ or $U_{\max}$. By allowing the un-quantified CTL until operator, we can write this query as follows:

$$\min E_h(true\ U\ (req \wedge \min E_g(true\ U_{\min}\ gr)))$$

The best response times from the requesting states s_1, s_3 and s_5 are respectively 5, 1, and 10. Therefore this query evaluates to 1 at s_0.

5.1 Syntax and Semantics

The formal syntax of this extension of Min-max CTL is as follows. B denotes the set of boolean formulas over the set of atomic propositions $q \in \mathcal{AP}$, S denotes CTL formulas, and Z denotes Min-max CTL formulas. C and C' are user defined functions.

- $B = false \mid true \mid q \mid \neg q \mid B \wedge B \mid B \vee B \mid \neg B$
- $S = B \mid S \wedge S \mid S \vee S \mid E(S\ U\ S) \mid A(S\ U\ S) \mid \neg S$
- $Q = \min \mid \max$
- $Z = Z \wedge S \mid QE_{C(g,h)}(S\ U_Q\ Z) \mid QE_{C'(g)}(S\ U_Q\ S) \mid QA_{C(g,h)}(S\ U_Q\ Z) \mid QA_{C'(g)}(S\ U_Q\ S) \mid QE_{C(g,h)}(S\ U\ Z) \mid QA_{C(g,h)}(S\ U\ Z) \mid QE_{C'(g)}(S\ U\ S) \mid QA_{C'(g)}(S\ U\ S)$

As in the basic Min-max CTL, we define $EvalS(f, s)$ to denote the value evaluated by the extended Min-max CTL state formula, f, on a state, s, of the timed model. Also, we use the function, $EvalP(f, \pi, C)$ to denote the value evaluated by the Min-max CTL path formula, f, on a path, π, of the timed model, with respect to the cost function, C.

The semantics of evaluation is similar to that of the basic Min-max CTL, except for the path formulas where we use the un-quantified CTL until operator. We present the semantics of evaluation for such path formulas. Let $(f = f_1\ U\ f_2)$. Since the un-quantified U operator has been used, the closing state, t, of a path π can be any state in π such that $t \models f_2$ and for each state w preceding t in π, $w \models f_1$. Thus, unlike in the case of the basic Min-max CTL, we may have multiple closing states on a path π. Therefore, the semantics of evaluation of a path formula on a path is defined in terms of the state which is chosen as the closing state.

The semantics of $EvalP(f, \pi, C)$ for a given closing state $t \in \pi$ is as follows:

- The g-value of π (with respect to t) is defined as the total delay from the starting state of π to state t. We

have two cases, namely if f_2 is a CTL formula, or if f_2 is a Min-max CTL formula.

Case-1: f_2 is CTL: In this case, by definition of Min-max CTL, the function C must be a function of the g-value of π. $EvalP(f, \pi, C)$ evaluates to the value of $C(g)$.

Case-2: f_2 is Min-max CTL: In this case, the function C is a function of the g-value of π and the h-value computed by $EvalS(f_2, t)$. $EvalP(f, \pi, C)$ evaluates to the value of $C(g, h)$.

The choice of the closing state depends on the quantifier, Q, preceding the E or A operator. For example, for formulas of the form $\min E_{C(g,h)}(f_1\ U\ f_2)$, the closing state which minimizes the value of $C(g, h)$ is the desired closing state.

6 Algorithm for Evaluation

In this section we provide an outline that the algorithm for basic Min-max CTL evaluation [10] can be extended for the evaluation of the extensions presented in Section 4 and Section 5. As in the basic Min-max CTL evaluation algorithm, we assume that the cost function, $C(g, h, k)$ is monotonic (increasing or decreasing) with respect to g, h and k.

By the semantics of the extension of Section 4, a state can be a *contributing* state in a path only if the path does not contain any state w preceding its closing state which returns a better value. For each state w, the algorithm considers those paths where w is the contributing state, and evaluates the best among them for each possible closing state. Monotonicity with respect to both h and k ensures the correctness of the value evaluated.

For the extension presented in Section 5, we need an algorithm to evaluate an extended Min-max CTL formula of the form $f = QE_C(f_1\ U\ f_2)$ in a state. Since the other Min-max CTL formulas are basic Min-max CTL formulas, their evaluation remains the same.

Definition 6.1 [f-path and f-cycle:]
A path, π, starting at a state s and going through a state s' is called a "f-path from s through s'" iff the state formula f holds in all states preceding s' in π. A f-cycle through a state t is a f-path from t through t.

A shortest length f-path from a state s through a state s' is one where s' occurs as early as in any other f-path from s through s'. The longest length f-path from s through s' is defined similarly, where s' occurs at least as late as any other f-path from s through s'. Obviously, a shortest length f-path will have no f-cycles. Hence shortest length f-paths can be found using any standard shortest path algorithms on the state transition graph in polynomial time [9].

The algorithm which we develop is a labeling algorithm. A state, s, in the timed model is labeled by a sub-formula, f, iff its CTL restriction is true in that state. Further, if the sub-formula is a Min-max CTL formula, then the evaluation algorithm augments the label with the value $EvalS(f, s)$. We develop the algorithm to evaluate the Min-max query for all the above extensions. We also prove the correctness of the algorithm and it is established that the algorithm runs in polynomial time. Due to the space constraint, we are not presenting the correctness proof of the algorithm in this paper. The algorithm for evaluation of complete Min-max CTL is presented in Appendix.

Lemma 6.1 *The complexity of finding the length of a shortest f-path or a longest f-path from a state s to a state t in a timed model is $O(|\mathcal{R}| + |S| \log |S|)$, where $|S|$ is the number of states in the model and $|\mathcal{R}|$ is the size of the transition relation $\mathcal{R}$.*

Proof: Each f-path from s to t includes only states which are labeled f, and the state t. We first remove from the transition graph those states (except t) which are not labeled f and the set of transitions to and from these states. This can be done in $O(|\mathcal{R}| + |S|)$ time. All paths in the reduced transition graph are f-paths.

Finding the shortest path between a pair of nodes in a graph with non-negative edge costs requires $O(|\mathcal{R}| + |S| \log |S|)$ time where $|\mathcal{R}|$ denotes the number of edges in the graph [9].

For determining the longest path length, we require to consider the cycles in the graph. If we find a path from s to t through a state j which is self-reachable (that is, j belongs to a f-cycle), then the longest path length from s to t is ∞. Otherwise, we use the algorithm for acyclic graphs. This can be achieved in $O(|\mathcal{R}| + |S|)$ time.

Theorem 6.1 *Algorithm Evaluate requires $O(|f|.|S|^3.(|\mathcal{R}| + |S| \log |S|))$ time to evaluate a Monotonic Min-max CTL formula f of length $|f|$ on a timed model $J = \langle \mathcal{AP}, S, \mathcal{R}, s_0, \mathcal{L} \rangle$*

Proof: Step 2.4 and Step 2.5 can be done by a single depth-first traversal in $O(|\mathcal{R}| + |S|)$ time. Step 2.6 requires us to determine whether states in H belong to any φ-cycle. Since the worst case number of states in H is $|S|$, this step can be completed in $O(|S|.(|\mathcal{R}| + |S|))$ time. By virtue of Lemma 6.1, the complexity of Step 2.7.1.2 and Step 2.9.1.1 is $O(|\mathcal{R}| + |S| \log |S|)$. Therefore, the complexity of evaluating the formula at one state is $O(|S|.(|\mathcal{R}| + |S| \log |S|))$ (when Step 2.7 or Step 2.9 requires) and the complexity of evaluating the formula at every state is given by $O(|S|^2.(|\mathcal{R}| + |S| \log |S|))$.

In Step 2.8, we are required to find f-paths from s to t *through a specific state* r. For a given s, t and r, the shortest

path from s to r combined with the shortest path from r to t yields a shortest path from s to t through r. Determining this requires $O(|\mathcal{R}| + |S| \log |S|)$ time. A similar argument applies to the longest path problem when paths through self reachable states are considered to be of length ∞ and we can determine the longest path from s to t through r in $O(|\mathcal{R}| + |S|)$ time (similar to Lemma 6.1). Since in the worst case, we may need to execute the algorithm for each choice of s, r and t, the total complexity of evaluating a formula at all states given the Min-max values of its subformulas at all states is $O(|S|^3.(|\mathcal{R}| + |S| \log |S|))$. By induction on the length of the formula, the complexity of Algorithm Evaluate is $O(|f|.|S|^3.(|\mathcal{R}| + |S| \log |S|))$.

7 Conclusion

Model checking with temporal logics such as TCTL, TLTL and RTCTL which explicitly reason about timing properties of transition systems is known to be more complex than reasoning about untimed temporal logics such as CTL. Specifically, CTL is attractive because it can be checked in time polynomial in the size of the transition system. Min-max CTL can be used to express interesting queries about the best case and worst case temporal behaviors of timed models. In this paper, we have extended Min-max CTL in three directions, exhibiting increase in expressive power in each case. Significantly, we have shown that each extension can still be evaluated in polynomial time thereby preserving the attractive computational complexity of Min-max CTL. After having these interesting extensions of basic Min-max CTL, the query language Min-max CTL becomes more expressive.

It is observed that a large class of the min/max type of path queries over graphs (like minimum cost path) which are polynomial time solvable can be expressed in Min-max CTL. However there are some graph properties where resultant graph is a subtree instead of a path of the original graph can not be expressed in Min-max CTL. One such example is finding the minimum cost spanning tree of a graph. Similarly the property like "what is the minimum distant to cover all nodes labeled with p" can not be expressed in Min-max CTL. Currently, we are looking for an extension of Min-max CTL to express such type of graph properties.

References

[1] R. Alur. Timed automata. *Manuscript: www.cis.upenn.edu/∼alur/Nato97.ps.gz*, 1998.

[2] R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[3] R. Alur and T. A. Henzinger. Real time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[4] R. Alur and T. A. Henzinger. A really temporal logic. *JACM*, 41(1):181–204, 1994.

[5] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transaction on Computer Aided Design*, 4(4):401–424, 1994.

[6] S. V. Campos, E. M. Clarke, W. Marrero, and M. Minea. Verus: A tool for quantitative analysis of finite-state real-time systems. *Workshop on Languages, Compilers and Tools for Real-Time Systems*, 1995.

[7] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transaction on Programming Languages & Systems*, 8(2):244–263, 1986.

[8] E. M. Clarke and R. P. Kurshan. Computer aided verification. *IEEE Spectrum*, 33(6):61–67, 1996.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, The MIT Press Cambridge, 1990.

[10] P. Dasgupta, P. P. Chakrabarti, J. K. Deka, and S. Sriram. Min-max computation tree logic. *Artificial Intelligence*, (127):137–162, March 2001.

[11] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. *First Annual Workshop on Computer-Aided Verification*, 1989.

[12] J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann Publisher Inc., 1980.

Appendix

The algorithm which we describe is a labeling algorithm. A state, s, in the timed model is labeled by a sub-formula, f, iff its CTL restriction is true in that state. Further, if the sub-formula is a Min-max CTL formula, then the evaluation algorithm augments the label with the value $EvalS(f, s)$.

Algorithm Evaluate(f,s)

1. Use CTL model checking techniques to label the states of the model with the sub-formulas of the CTL restriction of f. During this step, we ignore the delays on the transitions of the timed model.

2. The evaluation at a state, s, of a Min-max CTL formula, $f = QE_C(Q''[f_1]\ U_{Q'}\ f_2)$, where Q, Q' and Q'' are $\min$ or $\max$ quantifiers, is done as follows. *The procedure for evaluating formulas of the form $f = QA_C(Q''[f_1]\ U_{Q'}\ f_2)$ is exactly similar.*

   2.1 Recursively evaluate all Min-max CTL subformulas of f at all states of the model.

   2.2 If Q' is $\max$ then define $\varphi = f_1$ else define $\varphi = f_1 \wedge \neg f_2$.

   2.3 If $f = QE_C(f_1\ U\ f_2)$ then define $\varphi = f_1$

2.4 Let H denotes the set of states labeled f_2, which are reachable from s along φ-paths.

2.5 Let G denotes the set of states which belong to any φ-path from s to any state of H

2.6 If Q' is max then:
- 2.6.1 Remove each state n from H such that:
  - (a) n does not belong to any φ-cycle, and
  - (b) For each successor n' of n, $n' \models A(f_1 U f_2)$.
- 2.6.2 Label a state n from H with the symbol ∞ if:
  - (a) n belongs to a φ-cycle, and
  - (b) For each successor n' of n, $n' \models A(f_1 U f_2)$.

2.7 If $f = QE_C(f_1\ U_{Q'}\ f_2)$, then
- 2.7.1 For each state $t \in H$ do:
  - 2.7.1.1 If t is labeled ∞ then set $g = \infty$ and compute $W(t) = C(g,h)$, where $h = EvalS(f_2, t)$.
  - 2.7.1.2 Else Consult Table 1 to determine the required path type from s to t, determine the length g of that path, and compute $W(t) = C(g,h)$, where $h = EvalS(f_2, t)$.
- 2.7.2 If Q is min,
  set $EvalS(f,s) = \min\{W(t) | t \in H\}$
  else set $EvalS(f,s) = \max\{W(t) | t \in H\}$.

2.8 If $f = QE_C(Q''[f_1]\ U_{Q'}\ f_2)$, then
- 2.8.1 For each state $r \in G$ do:
  - 2.8.1.1 Let $l = EvalS(f_1, r)$
  - 2.8.1.2 If Q'' is max then define G_r as the set of states, w, such that $w \in G$ and $EvalS(f_1, w) \leq l$.
  - 2.8.1.3 If Q'' is min then define G_r as the set of states, w, such that $w \in G$ and $EvalS(f_1, w) \geq l$.
  - 2.8.1.4 For each state $t \in H$ do:
    - 2.8.1.4.1 Let $h = EvalS(f_2, t)$
    - 2.8.1.4.2 If there is no path from s to t through r and from s to t solely through states in G_r, Set $W(t) = null$
    - 2.8.1.4.3 Else
      - 2.8.1.4.3.1 If t is labeled ∞ then set $g = \infty$ and Compute $W(t) = C(g,h,l)$
      - 2.8.1.4.3.2 Else Consult Table 1 to determine the required path type from s to t through r and from s to t solely through states from G_r. Compute $W(t) = C(g,h,l)$ accordingly.
  - 2.8.1.5 If Q is min, then
    - 2.8.1.5.1 If $\forall t \in H, W(t) = null$, then set $W(t) = null$
    - 2.8.1.5.2 Otherwise, set $W(r) = \min\{W(t) \mid t \in H\ and\ W(t) \neq null\}$
  - 2.8.1.6 If Q is max, then
    - 2.8.1.6.1 If $\forall t \in H, W(t) = null$, then set $W(t) = null$
    - 2.8.1.6.2 Otherwise, set $W(r) = \max\{W(t) \mid t \in H\ and\ W(t) \neq null\}$
- 2.8.2 If Q is min,
  set $EvalS(f,s) = \min\{W(r) \mid r \in G\}$
  else set $EvalS(f,s) = \max\{W(r) \mid r \in G\}$

2.9 If $f = QE_C(f_1\ U\ f_2)$
- 2.9.1 For each state $t \in H$ do:
  - 2.9.1.1 Consult Table 2 to determine the required path type from s to t and compute $W(t) = C(g,h)$, where $h = EvalS(f_2, t)$.
- 2.9.2 If Q is min,
  set $EvalS(f,s) = \min\{W(t) \mid t \in H\}$
  else set $EvalS(f,s) = \max\{W(t) \mid t \in H\}$.

C-Type	Q-Type	Q'-Type	Best path type from s to t
Increasing	min	min	Shortest length $(f_1 \wedge \neg f_2)$-path
Increasing	min	max	Shortest length f_1-path
Increasing	max	min	Longest length $(f_1 \wedge \neg f_2)$-path
Increasing	max	max	Longest length f_1-path
Decreasing	min	min	Longest length $(f_1 \wedge \neg f_2)$-path
Decreasing	min	max	Longest length f_1-path
Decreasing	max	min	Shortest length $(f_1 \wedge \neg f_2)$-path
Decreasing	max	max	Shortest length f_1-path

Table 1. Best path types for $f = QE_C(f_1\ U_{Q'}\ f_2)$ or $f = QE_C(Q''[f_1]\ U_{Q'}\ f_2)$

C-Type	Q-Type	Best path type from s to t
Increasing	min	Shortest length f_1-path
Increasing	max	Longest length f_1-path
Decreasing	min	Longest length f_1-path
Decreasing	max	Shortest length f_1-path

Table 2. Best path types for $f = QE_C(f_1\ U\ f_2)$