Deterministic CTL Query Solving*

Marko Samer
Institute of Information Systems (DBAI)
Vienna University of Technology, Austria
samer@dbai.tuwien.ac.at

Helmut Veith
Institut für Informatik (I7)
Technische Universität München, Germany
veith@in.tum.de

Abstract

Temporal logic queries provide a natural framework to extend the realm of model checking from mere verification of engineers' specifications to computing previously unknown temporal properties of a system. Formally, temporal logic queries are patterns of temporal logic specifications which contain placeholders for subformulas; a solution to a temporal logic query is an instantiation which renders the specification true. In this paper, we investigate temporal logic queries that can be solved deterministically, i.e., solving such queries can be reduced in a deterministic manner to solving their subqueries at appropriate system states. We show that this kind of determinism is intimately related to the notion of intermediate collecting queries studied by the authors in previous work. We describe a large class of deterministically solvable CTL queries and devise a BDD-based symbolic algorithm for this class.

1 Introduction and Overview

Temporal logic query solving is an extension of model checking introduced in a seminal paper by Chan [2]. A temporal logic query is a temporal logic formula containing one or more occurrences of a distinguished proposition "?" which is treated as a placeholder. Given a system model $\mathfrak{K}$ and a query γ, a solution to γ in $\mathfrak{K}$ is a formula φ satisfying $\mathfrak{K} \models \gamma[\varphi]$. Thus, in contrast to a model checker which essentially returns a truth value and possibly a counterexample, the task of a query solver is to infer a set of formulas. Essentially, the query can be viewed as a "specification skeleton" which describes the solution space. Query solving provides a versatile framework which facilitates diverse verification tasks including legacy code analysis, counterexample understanding, and vacuity detection [2, 1, 5, 3, 4, 9].

The current paper focuses on the notion of *deterministic query solving*. We say that a query can be solved deterministically if solving the query can be reduced in a deterministic manner to solving its subqueries at appropriate system states. Note that deterministic query solving is *not* possible for arbitrary queries. The Kripke structure $\mathfrak{K}$ in Figure 1 illustrates deterministic query solving for the query $\gamma = \mathbf{A}(\varphi\,\mathbf{U}\,\mathbf{AG}\,?)$. This query γ has the following property: *A formula ψ is a solution to γ in $\mathfrak{K}$ if ψ is a solution to γ's immediate subquery $\bar{\gamma} = \mathbf{AG}\,?$ at a set $\mathcal{S}$ of states that is reachable from s_0 by going only through states at which φ holds.* This follows immediately from the semantics of the until operator $\mathbf{U}$. Examples of such sets are the set $\mathcal{S}_1 = \{s_2, s_4, s_5, s_8, s_9\}$ and the set $\mathcal{S}_2 = \{s_4, s_6, s_7, s_9, s_{10}, s_{11}, s_{12}\}$ in Figure 1. Thus, we know that $\mathcal{S}_1 \models \bar{\gamma}[\psi]$ as well as $\mathcal{S}_2 \models \bar{\gamma}[\psi]$ imply that ψ is a solution to γ in $\mathfrak{K}$. Conversely, if ψ is a solution to γ in $\mathfrak{K}$, we know that there exists some set $\mathcal{S}$ as above such that $\mathcal{S} \models \bar{\gamma}[\psi]$ but we do in general not know which one. Therefore, in a naive approach, to obtain all solutions to the query $\gamma = \mathbf{A}(\varphi\,\mathbf{U}\,\mathbf{AG}\,?)$ we would have to loop through all possibilities for $\mathcal{S}$.

For certain queries such as γ considered above, however, it is possible to determine a single set $\mathcal{S}$ which gives already all solutions. In this case we speak of deterministic query solving. In our example above,

*This work was jointly funded by the European network CoLogNET (IST-2001-33123) and the EU Network of Excellence REWERSE (506779). The results presented in this paper have been developed as part of [7].

Proceedings of the 12th International Symposium on Temporal Representation and Reasoning (TIME'05)
1530-1311/05 $20.00 © 2005 **IEEE**

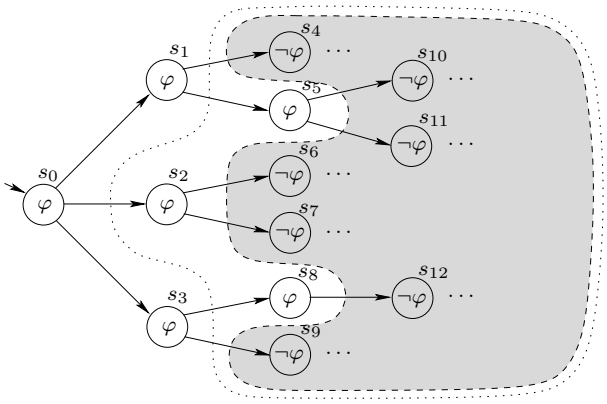

Figure 1. Example of solving $\mathbf{A}(\varphi\,\mathbf{U}\,\mathbf{AG}\,?)$

the set $\mathcal{S}_2$ is this set $\mathcal{S}$. To see this, note that all other possibilities for $\mathcal{S}$ "are located to the left of $\mathcal{S}_2$" in Figure 1. Moreover, the subquery $\bar{\gamma}$ is looking for an invariant. Since everything that is an invariant to the left of $\mathcal{S}_2$ must remain an invariant at $\mathcal{S}_2$, it suffices to compute the solutions at $\mathcal{S}_2$.

From this example we see that solving γ can be reduced to solving its subquery at a deterministically determined set of states. In this paper, we will investigate query solving algorithms which are based on such efficient deterministic reductions. Since deterministic reductions are not possible for all queries, the development of deterministic query solving algorithms first requires a (quite complicated) analysis of those queries which have the required property.

Our main results show that (i) deterministic query solving is closely related to the notion of intermediate collecting queries. Exploiting this relationship, we show (ii) how deterministic query solving gives rise to symbolic query solving algorithms. Our work achieves (iii) an extension of Chan's algorithm to a much wider class of queries.

In order to explain the results of this paper in more detail, we first need to review some of the previous research about temporal logic queries. In [2], Chan investigated CTL queries γ that are guaranteed to have an *exact solution* in every model, i.e., a solution ξ that implies all other solutions to γ. Chan presented a syntactic fragment of such queries and a symbolic BDD-based algorithm for solving them, albeit without proofs and with incorrect results. A systematic study by the authors aimed at an extension and correction of Chan's work by identifying *exact queries*, i.e., queries that have an exact solution if there exists any solution but that are not guaranteed to have a solution in every model. This research resulted in (i) the definition of an exact CTL query language which corrects the errors found in Chan's fragment [6, 8] and (ii) a syntactic characterization of exact LTL queries by a template grammar [7, 10]. Chan's symbolic algorithm is still poorly understood and has not been systematically investigated so far. Given the errors found in Chan's query language, and the absence of proofs in Chan's posthumous paper, the correctness and principles of his algorithm have remained unclear.

The current paper as a first result reports on a significant extension of the previously known class of exact CTL queries (cf. Table 1). The main result in this paper, however, concerns CTL *query solving algorithms*. Based on our insights on exact CTL queries, we give an exact and novel exposition of symbolic query solving algorithms and argue why the class of exact queries is amenable to classical BDD-based symbolic fixpoint algorithms. Since symbolic algorithms build up sets of states, they can only collect solutions, but not account for case distinctions. Consequently, reductions of solving queries to solving their subqueries can only be performed by symbolic algorithms when the set of states at which the subqueries have to be solved can be computed *deterministically*. This situation is intuitively accounted for in Chan's algorithm; his algorithm reduces the computation of solutions to a CTL query at a given set of states to the computation of solutions of a subquery at another set of states. This reduction is repeatedly applied until the placeholder is reached.

Our results also have the following intuitively appealing logical interpretation: Since exact queries are characterized by distributivity (i.e., $\gamma[\varphi] \wedge \gamma[\psi] \Leftrightarrow \gamma[\varphi \wedge \psi]$) [7, 10], it is natural to expect that distributivity holds in cases where the placeholder is universally quantified and is violated otherwise. Consequently, our results give an intuitive high-level explanation which fragments of LTL and CTL are exact.

This paper is organized as follows: In Section 2, we shortly summarize the formalisms used in the remainder of this paper. Afterwards, in Section 3, we formally introduce temporal logic queries and we present our syntactic fragment CTLQ^x of exact CTL queries. Then, we systematically investigate proper-

Proceedings of the 12th International Symposium on Temporal Representation and Reasoning (TIME'05)
1530-1311/05 $20.00 © 2005 **IEEE**

ties of queries in CTLQ^x in Section 4 consisting of two subsections. In Section 4.1, we show how nondeterminism in the sense of existential choices can be eliminated when solving queries in CTLQ^x. Based on these insights, we present our extension of Chan's symbolic algorithm and state its correctness in Section 4.2. Finally, we conclude in Section 5.

2 Preliminaries

We assume the reader is familiar with the basics of (symbolic) model checking, i.e., Kripke structures, the computation tree logic CTL based on the temporal operators $\mathbf{X}$ ("next"), $\mathbf{F}$ ("future"), $\mathbf{G}$ ("global"), and $\mathbf{U}$ ("until"), the least and greatest fixpoint-operators μ and ν, etc.

Let $\mathfrak{K} = (\mathcal{Q}, \mathcal{Q}_0, \Delta, \ell)$ be a *Kripke structure* over the set $\mathcal{A}$ of atomic propositions, where $\mathcal{Q}$ is a set of states, $\mathcal{Q}_0 \subseteq \mathcal{Q}$ is the set of initial states, $\Delta \subseteq \mathcal{Q} \times \mathcal{Q}$ is a total transition relation, and $\ell : \mathcal{Q} \to \wp(\mathcal{A})$ is a total labeling function. A *computation path* or simply *path* π in $\mathfrak{K}$ is an infinite sequence of states $\pi : \mathbb{N} \to \mathcal{Q}$ such that $(\pi(i), \pi(i+1)) \in \Delta$ for all $i \in \mathbb{N}$. We write π^n to denote the computation path satisfying $\pi^n(i) = \pi(n+i)$ for all $i \in \mathbb{N}$. As usual we write $\mathfrak{K}, s \models \varphi$ to denote that the CTL formula φ is satisfied at state s in $\mathfrak{K}$, and we write $\mathfrak{K} \models \varphi$ to denote $\mathfrak{K}, s_0 \models \varphi$, where s_0 is the initial state of $\mathfrak{K}$. For simplicity, we also write $\mathfrak{K}, \pi \models \varphi$ to denote $\mathfrak{K}, \pi(0) \models \varphi$ and $\mathfrak{K}, \mathcal{S} \models \varphi$ to denote $\mathfrak{K}, s \models \varphi$ for all $s \in \mathcal{S}$. We omit $\mathfrak{K}$ if it is clear from the context.

Following Chan [2], we use some additional temporal operators. In particular, we use the weak until operator $\varphi\,\mathbf{W}\,\psi \Leftrightarrow (\mathbf{G}\,\varphi) \vee (\varphi\,\mathbf{U}\,\psi)$. The other operators are variants of the strong until operator $\mathbf{U}$ and the weak until operator $\mathbf{W}$:

$$\varphi\,\mathring{\mathbf{U}}\,\psi \Leftrightarrow \varphi\,\mathbf{U}\,(\varphi \wedge \psi) \qquad \varphi\,\mathring{\mathbf{W}}\,\psi \Leftrightarrow \varphi\,\mathbf{W}\,(\varphi \wedge \psi)$$
$$\varphi\,\bar{\mathbf{U}}\,\psi \Leftrightarrow \varphi\,\mathbf{U}\,(\neg\varphi \wedge \psi) \qquad \varphi\,\bar{\mathbf{W}}\,\psi \Leftrightarrow \varphi\,\mathbf{W}\,(\neg\varphi \wedge \psi)$$

The additional operators will give rise to stronger temporal logic queries although they do not increase the expressive power of CTL. We call an n-ary temporal operator $\mathbf{O}$ monotonic in its k-th operand iff for all formulas $\theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_n$ it holds that $\varphi \Rightarrow \psi$ implies $\mathbf{O}(\theta_1, \ldots, \theta_{k-1}, \varphi, \theta_{k+1}, \ldots, \theta_n) \Rightarrow \mathbf{O}(\theta_1, \ldots, \theta_{k-1}, \psi, \theta_{k+1}, \ldots, \theta_n)$ for all formulas φ and ψ.

3 Exact CTL Queries

In this section, we survey some basic properties of temporal logic queries.

Definition 1 (CTL query)**.** A *CTL query* is a CTL formula where some subformulas are replaced by a special variable **?**, called *placeholder*. We write $\gamma[\varphi]$ to denote the result of substituting all occurrences of the placeholder in γ by φ. We denote the set of all CTL queries by CTLQ.

Definition 2 (Solution)**.** Let γ be a query, $\mathfrak{K}$ be a Kripke structure, and φ be a formula. If $\mathfrak{K} \models \gamma[\varphi]$, then we say that φ is a *solution* to γ in $\mathfrak{K}$. We denote the set of all solutions to γ in $\mathfrak{K}$ by $\mathrm{sol}(\mathfrak{K}, \gamma) = \{\varphi \mid \mathfrak{K} \models \gamma[\varphi]\}$. A solution ξ to a query γ in a Kripke structure $\mathfrak{K}$ is *exact* iff it holds that $\mathrm{sol}(\mathfrak{K}, \gamma) = \{\varphi \mid \xi \Rightarrow \varphi\}$.

We call a query γ *monotonic* iff $\varphi \Rightarrow \psi$ implies $\gamma[\varphi] \Rightarrow \gamma[\psi]$ for all formulas φ and ψ. Note that it follows immediately from this definition that γ (if monotonic) has a solution in $\mathfrak{K}$ iff $\mathfrak{K} \models \gamma[\top]$ and every formula is a solution to γ in $\mathfrak{K}$ iff $\mathfrak{K} \models \gamma[\bot]$. We are now able to give a formal definition of exact queries.

Definition 3 (Exact query)**.** A query is *exact* iff it has an exact solution in every Kripke structure where the set of solutions is not empty.

In analogy to Chan's EXPTIME-completeness proof for deciding validity of CTL queries [2] it can be shown by reduction from and to the validity of CTL formulas that deciding exactness of CTL queries is also EXPTIME-complete [7]. The main implication of this result is the fact that no simple grammar can recognize all exact CTL queries. Therefore, we define an extensive syntactic fragment CTLQ^x such that all queries in this fragment are exact. To this aim, consider the deterministic context-free template grammar in Table 1, where $\star$ is a special wildcard symbol representing any CTL formula. For example, the template $\mathbf{A}(\star\,\mathbf{U}\,\gamma)$ represents all CTL queries of the form $\mathbf{A}(\varphi\,\mathbf{U}\,\gamma)$, where φ is a CTL formula. In the following, we write CTLQ^1 for the language derived from non-terminal $\langle Q1\rangle$, CTLQ^2 for the language derived from $\langle Q2\rangle$, and so on. The language CTLQ^x is defined as $\bigcup_{i=1}^{10}$ CTLQ^i. Although the definition of CTLQ^x seems to be messy when considering the grammar in

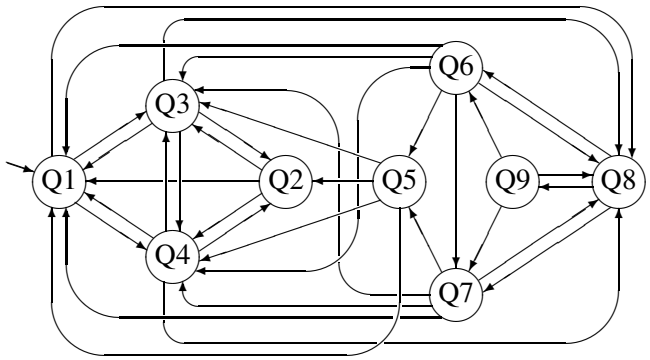

Figure 2. CTLQ dependence diagram

Table 1, it becomes very symmetric when considering the dependencies of the non-terminals as shown in Figure 2. Note that the vertex corresponding to non-terminal $\langle Q10 \rangle$ was omitted for simplicity because it has only incoming edges (from Q2, Q5, and Q9) but no outgoing edges.

By a series of nested inductive proofs, the following theorem can be shown.

Theorem 1 ([7]). *Every query in CTLQ^x is exact.*

The major complication in the proof arises from the fact that the dependencies between the sublanguages are circular. Therefore, we need to use the following auxiliary properties.

Definition 4. Let γ be a CTL query.

- We say γ is *strong collecting* iff:
  If $\pi \models \gamma[\varphi]$ and $\pi^n \models \gamma[\psi]$ for some $n \in \mathbb{N}$, then $\pi^n \models \gamma[\varphi \wedge \psi]$.
- We say γ is *boundary collecting* iff:
  If $\pi \models \gamma[\varphi]$ and $\pi^n \models \gamma[\psi]$ for some $n \in \mathbb{N}$, then $\pi^n \models \gamma[\varphi \wedge \psi]$ or $\pi \models \gamma[\bot]$.
- We say γ is *intermediate collecting* iff:
  If $\pi \models \gamma[\varphi]$ and $\pi^n \models \gamma[\psi]$ for some $n \in \mathbb{N}$, then $\pi^n \models \gamma[\varphi \wedge \psi]$ or there exists $r < n$ such that $\pi^r \models \gamma[\bot]$.
- We say γ is *weak collecting* iff:
  If $s \models \gamma[\varphi]$ and $s \models \gamma[\psi]$, then $s \models \gamma[\varphi \wedge \psi]$.

It follows immediately from Definition 4 that every strong collecting query is also boundary collecting, every boundary collecting query is also intermediate collecting ($r = 0$), and every intermediate collecting query is also weak collecting ($n = 0$). Therefore, we say a query is *collecting* iff it is at least *weak collecting*. The following result can be shown by inductive proofs on the structure and the number of queries in the sublanguages of CTLQ^x.

Lemma 1 ([7]). *Every query in CTLQ^1 and CTLQ^2 is weak collecting. Every query in CTLQ^3, CTLQ^4, and CTLQ^5 is intermediate collecting. Every query in CTLQ^6 and CTLQ^7 is boundary collecting. Every query in CTLQ^8, CTLQ^9, and CTLQ^{10} is strong collecting.*

Now, recall that CTLQ^x consists of the above CTL query languages, i.e., all queries in CTLQ^x are at least weak collecting. Moreover, it can be easily shown that all queries γ in CTLQ^x are monotonic. Hence, by the following theorem, we obtain Theorem 1 above.

Theorem 2 ([7, 10]). *A query is exact iff it is monotonic and collecting.*

Remark 1. In contrast to the characterization of LTLQ^x [7, 10], we are unfortunately not able to prove the maximality of CTLQ^x in the sense that all simple queries, i.e., queries whose subformulas are atomic and occur only once in the query, not in CTLQ^x are not exact. Quite the contrary, CTLQ^x is *not* maximal in this sense and therefore *not* a characterization of exact CTL queries. For example, consider the simple query $\gamma = \mathbf{AF}(a \wedge \mathbf{AF}(b \vee \mathbf{AG}\,?))$. It can be easily verified that $\gamma \notin$ CTLQ^x although γ is collecting. Finally, let us remark that a proof of maximality by counterexample construction as in the case of LTLQ^x is much more difficult in the case of CTLQ^x since a counterexample to the collecting property for queries in CTLQ^x is in general a computation tree instead of a computation path as in the case of LTLQ^x.

4 Solving Queries in CTLQ^x

In this section, we show how to symbolically compute an exact solution to queries in CTLQ^x. The connection between computing an exact solution and the collecting properties introduced in the previous section is: The collecting properties enable us to eliminate non-determinism in the sense of existential choices. In order to locate the cause of non-determinism in this

$\langle Q1\rangle$	::=	?	|	$\star \wedge \langle Q3\rangle$	|	$\star \wedge \langle Q4\rangle$	|	$\star \vee \langle Q2\rangle$	|
		$\mathbf{AX}\,\langle Q3\rangle$	|	$\mathbf{AX}\,\langle Q4\rangle$	|	$\mathbf{AX}\,\langle Q6\rangle$	|	$\mathbf{AX}\,\langle Q7\rangle$	|
		$\mathbf{A}(\langle Q3\rangle\,\mathring{\mathbf{U}}\,\star)$	|	$\mathbf{A}(\langle Q4\rangle\,\mathring{\mathbf{U}}\,\star)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q4\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q5\rangle)$	|
		$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q2\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q3\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q4\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q5\rangle)$	|
		$\mathbf{A}(\langle Q3\rangle\,\mathring{\mathbf{W}}\,\star)$	|	$\mathbf{A}(\langle Q4\rangle\,\mathring{\mathbf{W}}\,\star)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q2\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q3\rangle)$	|
		$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q4\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q5\rangle)$	|	$\star \wedge \langle Q1\rangle$	|	$\star \vee \langle Q1\rangle$	|
		$\mathbf{AX}\,\langle Q1\rangle$	|	$\mathbf{A}(\langle Q1\rangle\,\mathring{\mathbf{U}}\,\star)$	|	$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q1\rangle)$	|	$\mathbf{A}(\langle Q1\rangle\,\mathring{\mathbf{W}}\,\star)$	|
		$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q1\rangle)$	;						
$\langle Q2\rangle$	::=	$\star \wedge \langle Q5\rangle$	|	$\mathbf{AX}\,\langle Q5\rangle$	|	$\mathbf{A}(\langle Q5\rangle\,\mathring{\mathbf{U}}\,\star)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q3\rangle)$	|
		$\mathbf{A}(\langle Q5\rangle\,\mathring{\mathbf{W}}\,\star)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q3\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q4\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q5\rangle)$	|
		$\star \wedge \langle Q2\rangle$	|	$\mathbf{AX}\,\langle Q2\rangle$	|	$\mathbf{A}(\langle Q2\rangle\,\mathring{\mathbf{U}}\,\star)$	|	$\mathbf{A}(\langle Q2\rangle\,\mathring{\mathbf{W}}\,\star)$	;
$\langle Q3\rangle$	::=	$\mathbf{AF}\,\langle Q6\rangle$	|	$\mathbf{A}(\langle Q1\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\langle Q2\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\langle Q4\rangle\,\mathbf{U}\,\star)$	|
		$\mathbf{A}(\langle Q5\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\langle Q6\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\langle Q7\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\star\,\mathbf{U}\,\langle Q6\rangle)$	|
		$\star \vee \langle Q3\rangle$	|	$\mathbf{AF}\,\langle Q3\rangle$	|	$\mathbf{A}(\langle Q3\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\star\,\mathbf{U}\,\langle Q3\rangle)$	;
$\langle Q4\rangle$	::=	$\star \vee \langle Q5\rangle$	|	$\mathbf{AF}\,\langle Q5\rangle$	|	$\mathbf{AF}\,\langle Q7\rangle$	|	$\mathbf{A}(\langle Q6\rangle\,\mathring{\mathbf{U}}\,\star)$	|
		$\mathbf{A}(\langle Q7\rangle\,\mathring{\mathbf{U}}\,\star)$	|	$\mathbf{A}(\star\,\mathbf{U}\,\langle Q5\rangle)$	|	$\mathbf{A}(\star\,\mathbf{U}\,\langle Q7\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q7\rangle)$	|
		$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q6\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q7\rangle)$	|	$\mathbf{A}(\langle Q1\rangle\,\mathbf{W}\,\star)$	|	$\mathbf{A}(\langle Q2\rangle\,\mathbf{W}\,\star)$	|
		$\mathbf{A}(\langle Q3\rangle\,\mathbf{W}\,\star)$	|	$\mathbf{A}(\langle Q5\rangle\,\mathbf{W}\,\star)$	|	$\mathbf{A}(\langle Q6\rangle\,\mathbf{W}\,\star)$	|	$\mathbf{A}(\langle Q7\rangle\,\mathbf{W}\,\star)$	|
		$\mathbf{A}(\langle Q6\rangle\,\mathring{\mathbf{W}}\,\star)$	|	$\mathbf{A}(\langle Q7\rangle\,\mathring{\mathbf{W}}\,\star)$	|	$\mathbf{A}(\star\,\mathbf{W}\,\langle Q3\rangle)$	|	$\mathbf{A}(\star\,\mathbf{W}\,\langle Q5\rangle)$	|
		$\mathbf{A}(\star\,\mathbf{W}\,\langle Q6\rangle)$	|	$\mathbf{A}(\star\,\mathbf{W}\,\langle Q7\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q6\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q7\rangle)$	|
		$\star \vee \langle Q4\rangle$	|	$\mathbf{AF}\,\langle Q4\rangle$	|	$\mathbf{A}(\star\,\mathbf{U}\,\langle Q4\rangle)$	|	$\mathbf{A}(\langle Q4\rangle\,\mathbf{W}\,\star)$	|
		$\mathbf{A}(\star\,\mathbf{W}\,\langle Q4\rangle)$	;						
$\langle Q5\rangle$	::=	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q6\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q6\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q7\rangle)$	;		
$\langle Q6\rangle$	::=	$\mathbf{A}(\langle Q8\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\langle Q9\rangle\,\mathbf{U}\,\star)$	|	$\star \vee \langle Q6\rangle$	;		
$\langle Q7\rangle$	::=	$\star \wedge \langle Q6\rangle$	|	$\star \vee \langle Q8\rangle$	|	$\star \vee \langle Q9\rangle$	|	$\mathbf{A}(\langle Q8\rangle\,\mathbf{W}\,\star)$	|
		$\mathbf{A}(\langle Q9\rangle\,\mathbf{W}\,\star)$	|	$\star \wedge \langle Q7\rangle$	|	$\star \vee \langle Q7\rangle$	;		
$\langle Q8\rangle$	::=	$\mathbf{AF}\,\langle Q9\rangle$	|	$\mathbf{AG}\,\langle Q1\rangle$	|	$\mathbf{AG}\,\langle Q3\rangle$	|	$\mathbf{AG}\,\langle Q4\rangle$	|
		$\mathbf{AG}\,\langle Q6\rangle$	|	$\mathbf{AG}\,\langle Q7\rangle$	|	$\mathbf{A}(\star\,\mathbf{U}\,\langle Q9\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q9\rangle)$	|
		$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q9\rangle)$	|	$\mathbf{A}(\star\,\mathbf{W}\,\langle Q9\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q9\rangle)$	|	$\star \wedge \langle Q8\rangle$	|
		$\mathbf{AX}\,\langle Q8\rangle$	|	$\mathbf{AF}\,\langle Q8\rangle$	|	$\mathbf{AG}\,\langle Q8\rangle$	|	$\mathbf{A}(\langle Q8\rangle\,\mathring{\mathbf{U}}\,\star)$	|
		$\mathbf{A}(\star\,\mathbf{U}\,\langle Q8\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q8\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q8\rangle)$	|	$\mathbf{A}(\langle Q8\rangle\,\mathring{\mathbf{W}}\,\star)$	|
		$\mathbf{A}(\star\,\mathbf{W}\,\langle Q8\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q8\rangle)$	;				
$\langle Q9\rangle$	::=	$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q8\rangle)$	|	$\star \wedge \langle Q9\rangle$	|	$\mathbf{AX}\,\langle Q9\rangle$	|	$\mathbf{A}(\langle Q9\rangle\,\mathring{\mathbf{U}}\,\star)$	|
		$\mathbf{A}(\langle Q9\rangle\,\mathring{\mathbf{W}}\,\star)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q9\rangle)$	;				
$\langle Q10\rangle$	::=	$\mathbf{AG}\,\langle Q2\rangle$	|	$\mathbf{AG}\,\langle Q5\rangle$	|	$\mathbf{AG}\,\langle Q9\rangle$	|	$\star \wedge \langle Q10\rangle$	|
		$\star \vee \langle Q10\rangle$	|	$\mathbf{AX}\,\langle Q10\rangle$	|	$\mathbf{AF}\,\langle Q10\rangle$	|	$\mathbf{AG}\,\langle Q10\rangle$	|
		$\mathbf{A}(\langle Q10\rangle\,\mathbf{U}\,\star)$	|	$\mathbf{A}(\langle Q10\rangle\,\mathring{\mathbf{U}}\,\star)$	|	$\mathbf{A}(\star\,\mathbf{U}\,\langle Q10\rangle)$	|	$\mathbf{A}(\star\,\mathring{\mathbf{U}}\,\langle Q10\rangle)$	|
		$\mathbf{A}(\star\,\bar{\mathbf{U}}\,\langle Q10\rangle)$	|	$\mathbf{A}(\langle Q10\rangle\,\mathbf{W}\,\star)$	|	$\mathbf{A}(\langle Q10\rangle\,\mathring{\mathbf{W}}\,\star)$	|	$\mathbf{A}(\star\,\mathbf{W}\,\langle Q10\rangle)$	|
		$\mathbf{A}(\star\,\mathring{\mathbf{W}}\,\langle Q10\rangle)$	|	$\mathbf{A}(\star\,\bar{\mathbf{W}}\,\langle Q10\rangle)$	;				

Table 1. CTLQ^x production rules

context more systematically, note that temporal operators can be divided into *universal* and *existential* ones. Let us consider some examples.

Example 1. For every path π and formula φ it holds that $\pi \models \mathbf{G}\,\bar{\gamma}[\varphi]$ if and only if $\forall i \in \mathbb{N}.\ \pi^i \models \bar{\gamma}[\varphi]$. Thus, solving a query $\mathbf{G}\,\bar{\gamma}$ can be reduced to solving its subquery $\bar{\gamma}$ at universally quantified positions on a path. Hence, we classify the global operator $\mathbf{G}$ to be a universal operator.

In contrast, consider path π in Figure 3. Obviously, it holds that $\pi \models a\,\mathbf{U}\,\bar{\gamma}[\varphi]$ for every solution φ to $\gamma = a\,\mathbf{U}\,\bar{\gamma}$ on π, but solving γ cannot be reduced to solving its subquery $\bar{\gamma}$ at universally quantified positions on π. However, for every path π it holds that $\pi \models \theta\,\mathbf{U}\,\bar{\gamma}[\varphi]$ if and only if $\exists i \in \{j \in \mathbb{N} \mid j \leq n\}.\ \pi^i \models \bar{\gamma}[\varphi]$, where $n \in \mathbb{N}$ is the least number such that $\pi^n \not\models \theta$. Thus, solving a query $\theta\,\mathbf{U}\,\bar{\gamma}$ can be reduced to solving its subquery $\bar{\gamma}$ at existentially quantified positions on a path. Hence, we classify the strong until operator $\mathbf{U}$ with respect to its second argument to be an existential operator. Note, however, that the strong until operator with respect to its first argument is universal. To see this, let $n \in \mathbb{N}$ be the least number such that $\pi^n \models \theta$ for any path π and formula θ. Then, for every formula φ, it holds that $\pi \models \bar{\gamma}[\varphi]\,\mathbf{U}\,\theta$ if and only if $\forall i \in \{j \in \mathbb{N} \mid j < n\}.\ \pi^i \models \bar{\gamma}[\varphi]$.

The kind of non-determinism we consider in this section arises from existential choices when solving a query top-down by a reduction to solving its subqueries as demonstrated in Example 1. The formal starting point of our investigations is therefore:

Definition 5 (Universal, Existential)**.** Let $\mathbf{O}$ be an n-ary temporal operator that is monotonic in its k-th operand. Then, we define $\mathbf{O}$ to be *universal* with respect to its k-th operand iff for all paths π and formulas $\psi_1, \ldots, \psi_{k-1}, \psi_{k+1}, \ldots, \psi_n$ satisfying $\pi \models \mathbf{O}(\psi_1, \ldots, \psi_{k-1}, \top, \psi_{k+1}, \ldots, \psi_n)$ there exists a set $\mathcal{I} \subseteq \mathbb{N}$, called *reduction set*, such that $\pi \models \mathbf{O}(\psi_1, \ldots, \psi_{k-1}, \varphi, \psi_{k+1}, \ldots, \psi_n)$ iff $\forall i \in \mathcal{I}.\, \pi^i \models \varphi$. We define $\mathbf{O}$ to be *existential* with respect to its k-th operand iff $\mathbf{O}$ is not universal with respect to its k-th operand and for all paths π and formulas $\psi_1, \ldots, \psi_{k-1}, \psi_{k+1}, \ldots, \psi_n$ satisfying $\pi \not\models \mathbf{O}(\psi_1, \ldots, \psi_{k-1}, \bot, \psi_{k+1}, \ldots, \psi_n)$ there exists a set $\mathcal{I} \subseteq \mathbb{N}$, called *reduction set*, such that $\pi \models \mathbf{O}(\psi_1, \ldots, \psi_{k-1}, \varphi, \psi_{k+1}, \ldots, \psi_n)$ iff $\exists i \in \mathcal{I}.\, \pi^i \models \varphi$. If $\mathbf{O}$ is an n-ary operator that is universal (resp. existential) with respect to its k-th operand and $\mathbf{O}(\psi_1, \ldots, \psi_{k-1}, \bar{\gamma}, \psi_{k+1}, \ldots, \psi_n)$ occurs in a query γ for any formulas $\psi_1, \ldots, \psi_{k-1}, \psi_{k+1}, \ldots, \psi_n$, then we say that $\bar{\gamma}$ is a *universally* (resp. *existentially*) occurring subquery of γ.

Note that all temporal operators used in this paper are either universal or existential with respect to a selected operand. This fact is summarized in Table 2, where the placeholder indicates the corresponding operand. For example, consider the query $\gamma = \mathbf{AG}(a \vee \mathbf{A}(b\,\mathbf{U}\,\mathbf{AX}\,\bar{\gamma}))$. Then, $\mathbf{AX}\,\bar{\gamma}$ is existentially occurring in γ, whereas $\bar{\gamma}$ is universally occurring in γ.

Remark 2. Note that the restrictions to paths on which $\mathbf{O}(\psi_1, \ldots, \psi_{k-1}, \top, \psi_{k+1}, \ldots, \psi_n)$ holds and $\mathbf{O}(\psi_1, \ldots, \psi_{k-1}, \bot, \psi_{k+1}, \ldots, \psi_n)$ does not hold respectively in the above definition are necessary in order to achieve our desired classification. For example, the strong until operator $\mathbf{U}$ with respect to its first argument would not be universal if we omitted the first condition, since it would not be universal on paths that do not satisfy its second argument at any state. On the other hand, the weak until operator $\mathbf{W}$ with respect to its second argument would not be existential if we omitted the second condition, since it would not be existential on paths that satisfy its first argument globally.

The following definition will enable us to easily describe an appropriate reduction set $\mathcal{I}$ according to Definition 5 for the existential operators in Table 2.

Definition 6 (Prefix indices)**.** A set $\mathcal{I} \subseteq \mathbb{N}$ of natural numbers is a set of *prefix indices* iff for each $n \in \mathcal{I}$ it holds that for all $i < n$, $i \in \mathcal{I}$. In particular, for any path π and formula φ, we define the set of prefix indices $\mathcal{I}_\pi(\varphi) = \{n \in \mathbb{N} \mid \forall i < n.\ \pi^i \models \varphi\}$.

Note that a set of prefix indices is either an initial segment of $\mathbb{N}$ (i.e., a set of the form $\{i \in \mathbb{N} \mid i \leq n\}$ for some $n \in \mathbb{N}$) or $\mathbb{N}$ itself. For example, let π be the path shown in Figure 3. Then, $\mathcal{I}_\pi(a) = \{0, 1, 2\}$, $\mathcal{I}_\pi(b) = \{0\}$, and $\mathcal{I}_\pi(a \vee b) = \mathbb{N}$. The following lemma shows that for the strong until operator with respect to its second argument, the set of prefix indices is an appropriate reduction set according to Definition 5. It follows immediately from the definition of the strong until operator.

Proceedings of the 12th International Symposium on Temporal Representation and Reasoning (TIME'05)
1530-1311/05 $20.00 © 2005 **IEEE**

Universal				Existential			
$\mathbf{X}?$	$?\,\mathbf{U}\,\varphi$	$?\,\mathring{\mathbf{U}}\,\varphi$	$\varphi\,\bar{\mathbf{U}}?$	$\mathbf{F}?$			
$\mathbf{G}?$	$?\,\mathbf{W}\,\varphi$	$?\,\mathring{\mathbf{W}}\,\varphi$	$\varphi\,\bar{\mathbf{W}}?$	$\varphi\,\mathbf{U}?$	$\varphi\,\mathring{\mathbf{U}}?$	$\varphi\,\mathbf{W}?$	$\varphi\,\mathring{\mathbf{W}}?$

Table 2. Classification of temporal operators

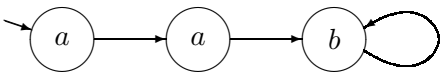

Figure 3. Prefix index example

Lemma 2. *For every path π and formula ψ it holds that $\pi \models \psi\,\mathbf{U}\,\varphi$ iff $\exists i \in \mathcal{I}_\pi(\psi).\ \pi^i \models \varphi$.*

Example 2. Consider path π shown in Figure 3 and query $\gamma = a\,\mathbf{U}\,\bar{\gamma}$. Assume that we want to solve γ on π by reducing it to solving $\bar{\gamma}$ on π. Since the strong until operator with respect to its second argument is an existential operator (cf. Table 2), i.e., $\bar{\gamma}$ is an existentially occurring subquery, we know that there exists a reduction of the form $\pi \models \gamma[\varphi]$ iff $\exists i \in \mathcal{I}.\ \pi^i \models \bar{\gamma}[\varphi]$ for some set $\mathcal{I} \subseteq \mathbb{N}$. By Lemma 2, we are allowed to choose $\mathcal{I} = \mathcal{I}_\pi(a) = \{0, 1, 2\}$. Hence, we know that solving γ on π can be reduced to solving $\bar{\gamma}$ at states with indices in $\mathcal{I}$, i.e., we obtain all solutions to γ on π by computing the solutions to $\bar{\gamma}$ on π^0, π^1, and π^2.

Note that all existential operators in Table 2 are variants of the strong until operator. Thus, as we will see in Section 4.2, it suffices to have an appropriate reduction set for the strong until operator $\mathbf{U}$.

Remark 3. Not surprisingly, there is a close relationship between the classification into universal and existential temporal operators and the first-order quantification of the corresponding operands in the definition of their semantics. However, there are also universal operators whose corresponding operand is existentially quantified, e.g., the disjoint strong until operator $\bar{\mathbf{U}}$ with respect to its second operand. But in these cases it is easy to see that the semantics of the operators can be equivalently redefined by using the uniqueness quantifier $\exists!$ instead of the existential quantifier. Consequently, since the reduction set in such cases is a singleton set, universal and existential quantification over this set would be equivalent. However, since universal quantification has a higher priority in Definition 5, we obtain our desired classification.

4.1 Eliminating Non-determinism

The following fact enables us to build up on the auxiliary results of our exactness proof in the previous section. It can be easily verified by the grammar defining CTLQ^x in Table 1.

Proposition 1. *All existentially occurring subqueries of queries in our fragment CTLQ^x are intermediate collecting.*

Intuitively, this means whenever in a CTLQ^x query we meet an operator which has an existential semantics (such as $\varphi\,\mathbf{U}\,?$), then its context in the CTLQ^x grammar ensures that a deterministic reduction is possible. This follows immediately from the intermediate collecting property and allows an algorithmic interpretation in order to eliminate existential choices as described below.

Lemma 3 ([7])**.** *Let γ be an intermediate collecting query and π be a path. Suppose that $\pi^n \models \gamma[\top]$ for some $n \in \mathbb{N}$. If $\pi^i \not\models \gamma[\bot]$ for all $i < n$, then every solution to γ on π is a solution to γ on π^n.*

In other words, it suffices to perform several simple *model checking* calls to achieve a deterministic reduction for *query solving*, and thus we can alleviate the problem posed by the existential quantifier. Since it takes one call to a symbolic model checker to verify $\gamma[\bot]$ for all states, the complexity of the model checking step is significantly lower than it were for query solving. Based on this principle, we describe a symbolic algorithm which computes exact solutions for all queries in CTLQ^x. Chan's algorithm then can be obtained as a special case of our algorithm.

Since solving an existentially occurring query has in general to be done at several states on a path, we need the following definition.

Proceedings of the 12th International Symposium on Temporal Representation and Reasoning (TIME'05)
1530-1311/05 $20.00 © 2005 **IEEE**

Definition 7. Let γ be a query, π be a path, and $\mathcal{I} \subseteq \mathbb{N}$. Then, we define the set of solutions to γ at positions in $\mathcal{I}$ on π by $\mathrm{sol}_{\mathcal{I}}(\pi, \gamma) = \bigcup_{i \in \mathcal{I}} \{\varphi \mid \pi^i \models \gamma[\varphi]\}$.

For our purposes, the set $\mathcal{I}$ will be a set of prefix indices representing the existential choices of positions on π when solving an existentially occurring subquery γ. Thus, in order to obtain *all* solutions, γ has in general to be solved at all states with indices in $\mathcal{I}$. Note that if there exists $i \in \mathcal{I}$ such that $\pi^i \models \gamma[\bot]$, then, if γ is monotonic, *every* formula is an element of $\mathrm{sol}_{\mathcal{I}}(\pi, \gamma)$ and therefore the solutions at other states with indices in $\mathcal{I}$ do not affect $\mathrm{sol}_{\mathcal{I}}(\pi, \gamma)$. This case is somehow exceptional since it can be simply checked by evaluating the formula $\gamma[\bot]$ at all states with indices in $\mathcal{I}$, which can be performed by a single *symbolic* model checking call.

Otherwise, if no such a prefix index exists, it follows immediately from the intermediate collecting property by repeated application of Lemma 3 that

$$\mathrm{sol}_{\{i_0\}}(\pi, \gamma) \subseteq \mathrm{sol}_{\{i_1\}}(\pi, \gamma) \subseteq \mathrm{sol}_{\{i_2\}}(\pi, \gamma) \subseteq \cdots,$$

where $i_0 < i_1 < i_2 < \cdots$ and $\pi^n \models \gamma[\top]$ for all $n \in \{i_1, i_2, i_3, \ldots\} \subseteq \mathcal{I}$. Hence, when solving an intermediate collecting query at several states on a path, it suffices to solve the query at states with indices as high as possible. In particular, if there exists a highest index $n \in \mathcal{I}$ such that $\pi^n \models \gamma[\top]$, it suffices to solve the query at this state.

Example 3. Recall Example 2 and let us now consider the case where $\bar{\gamma}$ is intermediate collecting. In this case we can determinize the reduction in such a way that $\mathcal{I}$ becomes a singleton set. In particular, if $\pi^i \models \bar{\gamma}[\top]$ and $\pi^i \not\models \bar{\gamma}[\bot]$ for all $i \in \mathcal{I}_\pi(a)$, it follows from the intermediate collecting property according to Lemma 3 that $\mathrm{sol}_{\{0\}}(\pi, \bar{\gamma}) \subseteq \mathrm{sol}_{\{1\}}(\pi, \bar{\gamma}) \subseteq \mathrm{sol}_{\{2\}}(\pi, \bar{\gamma})$. Thus, the solutions obtained by solving $\bar{\gamma}$ on π^0 and π^1 are also solutions on π^2. Hence, it suffices to solve $\bar{\gamma}$ on π^2, i.e., the state with highest index in $\mathcal{I}_\pi(a)$. So we know that $\pi \models \gamma[\varphi]$ iff $\pi^2 \models \bar{\gamma}[\varphi]$ for all φ.

We are now going to state this insight formally which will enable us to extend the Chan algorithm and can be used in order to prove its correctness.

Lemma 4 ([7])**.** *Let γ be an intermediate collecting query, π be a path, and $\mathcal{I} \subseteq \mathbb{N}$. Suppose that there is a least index $m \in \mathcal{I}$ and a highest index $n \in \mathcal{I}$ such that $\pi^m \models \gamma[\top]$ and $\pi^n \models \gamma[\top]$. If $\pi^i \not\models \gamma[\bot]$ for all $m \leq i < n$, then $\mathrm{sol}_{\{n\}}(\pi, \gamma) = \mathrm{sol}_{\mathcal{I}}(\pi, \gamma)$.*

Note that Lemma 4 does not cover all cases that may appear when solving a query. In particular, it says nothing about the case where the highest index $n \in \mathcal{I}$ does not exist. This case occurs if $\mathcal{I}$ contains an infinite number of indices i satisfying $\pi^i \models \gamma[\top]$. Note that such indices must refer to states in a cycle, since only states in a cycle have an infinite number of indices. For every path π, let us define the set of *cycle indices* $\mathrm{cycle}(\pi) = \{i \in \mathbb{N} \mid i \geq k\}$, where k is the length of the path prefix of π before the first cycle starts. Thus, we are able to state the following lemma.

Lemma 5 ([7])**.** *Let γ be an intermediate collecting query, π be a path, and $\mathcal{I} \subseteq \mathbb{N}$. Suppose that there is a least index $m \in \mathcal{I}$ and an index $n \in \mathrm{cycle}(\pi) \cap \mathcal{I}$ such that $\pi^m \models \gamma[\top]$ and $\pi^n \models \gamma[\top]$. If $\pi^i \not\models \gamma[\bot]$ for all $i \geq m$, then $\mathrm{sol}_{\{n\}}(\pi, \gamma) = \mathrm{sol}_{\mathcal{I}}(\pi, \gamma)$.*

4.2 The Extended Chan Algorithm

The Chan algorithm was introduced by William Chan [2] in order to solve queries in his syntactic fragment of *valid* CTL queries, i.e., CTL queries that are guaranteed to have an exact solution in every model. However, Chan neither proved the correctness of his algorithm nor did he describe its functionality. With our insights above, it is possible to prove the correctness of Chan's algorithm when applied to valid queries within our fragment CTLQ^x. Note that such queries can be simply obtained by restricting the grammar in Table 1 to those operators that guarantee validity.

In the following, we are interested in a generalization of Chan's algorithm that is able to solve queries in the whole fragment CTLQ^x, i.e., queries that have an exact solution if there exists any solution.

Definition 8 (Auxiliary sets)**.** Following Chan, we introduce the following three macros (parameterized by φ and γ) as abbreviations:

$$\begin{aligned}
\mathcal{R}_\varphi &= \mu\mathcal{Z}.((\mathcal{Q} \cup \mathrm{post}_\exists(\mathcal{Z})) \cap [\![\varphi]\!]) \\
\mathcal{C}^\gamma_\varphi &= \nu\mathcal{Z}.(\mathcal{R}_{\varphi \wedge \neg\gamma[\bot]} \cap \mathrm{post}_\exists(\mathcal{Z})) \\
\mathcal{B}^\gamma_\varphi &= (\mathcal{Q} \cup \mathrm{post}_\exists(\mathcal{R}_{\varphi \wedge \neg\gamma[\bot]})) \setminus ([\![\varphi]\!] \cup [\![\gamma[\bot]]\!])
\end{aligned}$$

The intuitive meaning of these three auxiliary sets is illustrated in Figure 4, where the initial set $\mathcal{Q}$ is assumed to consist of the four double-circled states. The

Algorithm 1 The extended Chan algorithm

function $\mathrm{FSol}(\gamma, \varphi, \mathcal{Q}, \mathit{cycle})$ **begin**
  if cycle **then** $\mathcal{C} = \mathcal{C}^\gamma_\varphi$ **else** $\mathcal{C} = \emptyset$;
  $\mathcal{U}_1 = \nu\mathcal{Z}.((\mathcal{C} \setminus [\![\gamma[\top]]\!]) \cap \mathrm{post}_\exists(\mathcal{Z}))$;
  $\mathcal{U}_2 = \mathcal{U}_1 \cup (\mathcal{B}^\gamma_\varphi \setminus [\![\gamma[\top]]\!])$;
  $\mathcal{U}_3 = \mu\mathcal{Z}.(((\mathcal{U}_2 \cup \mathrm{pre}_\exists(\mathcal{Z})) \cap \mathcal{R}_{\varphi\wedge\neg\gamma[\bot]}) \setminus [\![\gamma[\top]]\!])$;
  ret $((\mathrm{pre}_\exists(\mathcal{U}_3) \cap \mathcal{R}_{\varphi\wedge\neg\gamma[\bot]}) \cup \mathcal{B}^\gamma_\varphi \cup \mathcal{C}) \cap [\![\gamma[\top]]\!]$;
end

function $\mathrm{ESol}(\gamma, \mathcal{Q})$ **begin**
  case γ **of**
    ?: **return** $\mathcal{Q}$;
    $\theta \wedge \bar\gamma$**:** **return** $\mathrm{ESol}(\bar\gamma, \mathcal{Q})$;
    $\theta \vee \bar\gamma$**:** **return** $\mathrm{ESol}(\bar\gamma, \mathcal{Q} \setminus [\![\theta]\!])$;
    $\mathbf{AX}\,\bar\gamma$**:** **return** $\mathrm{ESol}(\bar\gamma, \mathrm{post}_\exists(\mathcal{Q}))$;
    $\mathbf{AF}\,\bar\gamma$**:** **return** $\mathrm{ESol}(\mathbf{A}(\top\,\mathbf{U}\,\bar\gamma), \mathcal{Q})$;
    $\mathbf{AG}\,\bar\gamma$**:** **return** $\mathrm{ESol}(\mathbf{A}(\bar\gamma\,\mathring{\mathbf{W}}\,\bot), \mathcal{Q})$;
    $\mathbf{A}(\bar\gamma\,\mathbf{U}\,\theta)$**:** **return** $\mathrm{ESol}(\mathbf{A}((\theta \vee \bar\gamma)\,\mathring{\mathbf{W}}\,\theta), \mathcal{Q})$;
    $\mathbf{A}(\bar\gamma\,\mathring{\mathbf{U}}\,\theta)$**:** **return** $\mathrm{ESol}(\mathbf{A}(\bar\gamma\,\mathring{\mathbf{W}}\,\theta), \mathcal{Q})$;
    $\mathbf{A}(\theta\,\mathbf{U}\,\bar\gamma)$**:** **return** $\mathrm{ESol}(\bar\gamma, \mathrm{FSol}(\bar\gamma, \theta, \mathcal{Q}, \mathit{true}))$;
    $\mathbf{A}(\theta\,\mathring{\mathbf{U}}\,\bar\gamma)$**:** **return** $\mathrm{ESol}(\mathbf{A}(\theta\,\mathbf{U}\,(\theta \wedge \bar\gamma)), \mathcal{Q})$;
    $\mathbf{A}(\theta\,\bar{\mathbf{U}}\,\bar\gamma)$**:** **return** $\mathrm{ESol}(\mathbf{A}(\theta\,\bar{\mathbf{W}}\,\bar\gamma), \mathcal{Q})$;
    $\mathbf{A}(\bar\gamma\,\mathbf{W}\,\theta)$**:** **return** $\mathrm{ESol}(\mathbf{A}((\theta \vee \bar\gamma)\,\mathring{\mathbf{W}}\,\theta), \mathcal{Q})$;
    $\mathbf{A}(\bar\gamma\,\mathring{\mathbf{W}}\,\theta)$**:** **return** $\mathrm{ESol}(\bar\gamma, \mathcal{Q} \cup \mathrm{post}_\exists(\mathcal{R}_{\neg\theta}))$;
    $\mathbf{A}(\theta\,\mathbf{W}\,\bar\gamma)$**:** **return** $\mathrm{ESol}(\bar\gamma, \mathrm{FSol}(\bar\gamma, \theta, \mathcal{Q}, \mathit{false}))$;
    $\mathbf{A}(\theta\,\mathring{\mathbf{W}}\,\bar\gamma)$**:** **return** $\mathrm{ESol}(\mathbf{A}(\theta\,\mathbf{W}\,(\theta \wedge \bar\gamma)), \mathcal{Q})$;
    $\mathbf{A}(\theta\,\bar{\mathbf{W}}\,\bar\gamma)$**:** **return** $\mathrm{ESol}(\bar\gamma, (\mathcal{Q} \cup \mathrm{post}_\exists(\mathcal{R}_\theta)) \setminus [\![\theta]\!])$;
  esac
end

set $\mathcal{R}_{\varphi\wedge\neg\gamma[\bot]}$ consists of those states that are reachable from states in $\mathcal{Q}$ by going only through states at which $\varphi\wedge\neg\gamma[\bot]$ holds. In particular, $\mathcal{R}_\top$ consists of all states that are reachable from states in $\mathcal{Q}$. The set $\mathcal{C}^\gamma_\varphi$ consists of all states within a cycle in $\mathcal{R}_{\varphi\wedge\neg\gamma[\bot]}$. Finally, the set $\mathcal{B}^\gamma_\varphi$ consists of the boundary of $\mathcal{R}_{\varphi\wedge\neg\gamma[\bot]}$, i.e., the first states on each path starting from $\mathcal{Q}$ that are not in $\mathcal{R}_{\varphi\wedge\neg\gamma[\bot]}$ and at which $\gamma[\bot]$ does not hold. The following lemma states the meaning of these sets more formally.

Lemma 6 ([7])**.** *Let γ be a query, φ be a formula, and $\mathcal{Q}$ be a set of states in a Kripke structure. Moreover, let $\Pi \subseteq \mathrm{paths}(\mathcal{Q})$ such that $\pi \in \Pi$ iff $\pi^i \not\models \gamma[\bot]$ for all $i \in \mathcal{I}_\pi(\varphi)$. Then, $\mathcal{R}_\varphi$, $\mathcal{C}^\gamma_\varphi$, and $\mathcal{B}^\gamma_\varphi$ are the sets of states on paths $\pi \in \mathrm{paths}(\mathcal{Q})$ such that*

1. *$\pi(n) \in \mathcal{R}_\varphi$ iff for all $i \le n$ it holds that $\pi^i \models \varphi$.*
2. *$\pi(n) \in \mathcal{C}^\gamma_\varphi$ iff $\pi \in \Pi$, $\mathcal{I}_\pi(\varphi)$ is infinite, and $n \in \mathrm{cycle}(\pi)$.*
3. *$\pi(n) \in \mathcal{B}^\gamma_\varphi$ iff $\pi \in \Pi$, $\mathcal{I}_\pi(\varphi)$ is finite, and $n = \max(\mathcal{I}_\pi(\varphi))$.*

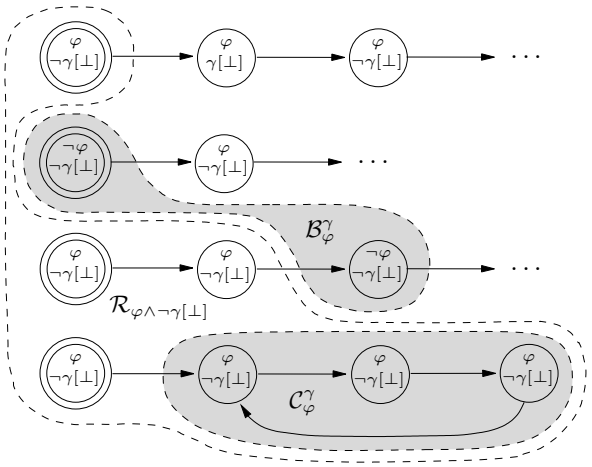

Figure 4. Auxiliary sets in Chan's algorithm

It is thus easy to see that the states in $\mathcal{B}_\varphi$ are those with the highest index on each path satisfying the conditions of Lemma 4, and the states in $\mathcal{C}_\varphi$ are those within a cycle on each path satisfying the conditions of Lemma 5. Hence, according to Lemma 4 and Lemma 5, solving a query in our fragment can be reduced to solving its existentially occurring valid subqueries at the set of states $\mathcal{B}_\varphi$ and $\mathcal{C}_\varphi$ without existential choices. Although this kind of determinization is not possible in general, it can be extended to the whole fragment CTLQ^x. However, when extending Chan's algorithm (see Algorithm 1) to the whole fragment CTLQ^x, the existence of a solution at each state is no longer guaranteed. Therefore, the required set of states satisfying the conditions of Lemma 4 and Lemma 5 such that solving a query can be reduced to solving its subqueries at this set, has to be computed in a more sophisticated way. In particular, we compute the set of states that correspond to highest indices resp. cycle indices in $\mathcal{I}$. Afterwards, we traverse the corresponding paths backwards until a state on each path is found at which γ has a solution. These are then the states with highest indices among the states at which γ has a solution, i.e., it suffices to solve γ at this uniquely determined set of states. This idea of computing the states at which γ has a solution and that are furthest away is implemented by the function FSol in Algorithm 1. Intuitively, the sets $\mathcal{U}_1$ and $\mathcal{U}_2$ together consist of all states with highest indices but on which γ has no solution. The set $\mathcal{U}_3$ consists then of all states that can be reached by going backwards as long as γ does not have a solution. Thus, by making a further

step backwards, we obtain the desired set of states, i.e., the states with highest indices at which γ has a solution. In this way, it is possible to prove the correctness of the extended Chan algorithm.

Definition 9 (Solution states)**.** Let γ be a query and $\mathcal{Q}$ be a set of states in a Kripke structure. A set of states $\mathcal{S}$ is the unique set of *solution states* to γ at $\mathcal{Q}$ iff it holds that $\mathcal{S} \models \varphi$ iff $\mathcal{Q} \models \gamma[\varphi]$ for all formulas φ.

The following theorem states our main result. It can be proved by structural induction on γ.

Theorem 3 ([7])**.** *Let $\gamma \in$ CTLQ^x and $\mathcal{Q}$ be a set of states in a Kripke structure. Moreover, let* ESol *be the function defined in Algorithm 1. Suppose that $\mathcal{Q} \models \gamma[\top]$, i.e., γ has a solution at each state in $\mathcal{Q}$. Then,* $\mathrm{ESol}(\gamma, \mathcal{Q})$ *returns the unique set of solution states to γ at $\mathcal{Q}$.*

In particular, the following corollary shows how to obtain a propositional exact solution by the extended Chan algorithm.

Corollary 1 ([7])**.** *Let $\gamma \in$ CTLQ^x and s_0 be the initial state of a Kripke structure $\mathfrak{K}$. Moreover, let* ESol *be the function defined in Algorithm 1. Then, the characteristic function of the set returned by* $\mathrm{ESol}(\gamma, \{s_0\})$ *is a propositional exact solution to γ in $\mathfrak{K}$.*

5 Conclusion

In this paper we have systematically investigated symbolic algorithms for query solving and have argued that symbolic algorithms work for queries which can be solved by a recursive deterministic descent.

To conclude this paper, let us consider the question of exact queries and their relevance. It is evident that temporal logic queries will in many cases have multiple incomparable solutions; algorithms for computing multiple solutions as well as extensions to multiple placeholders have been the subject of important related work [1, 5, 3, 4]. However, the only existing implementation TLQSolver [3] we are aware of is based on a multi-valued model checker. Since the multi-valued setting is essential to their approach, we believe that the results and methods in the current paper are orthogonal to the multi-valued approach. We speculate that the central role of determinism in the Chan style approach in combination with the highly optimized environment of tools such as SMV will incur algorithmic advantages, leading to a trade-off in our case between lower expressibility and higher expected performance. A final assertion on the practical performance of both approaches can of course only be achieved by systematic experiments. Therefore, natural future work includes the implementation and experimental evaluation of our algorithm.

References

[1] G. Bruns and P. Godefroid. Temporal logic query checking. In *Proc. of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 409–417. IEEE Computer Society, 2001.

[2] W. Chan. Temporal-logic queries. In *Proc. of the 12th International Conference on Computer Aided Verification (CAV)*, volume 1855 of *LNCS*, pages 450–463. Springer-Verlag, 2000.

[3] M. Chechik and A. Gurfinkel. TLQSolver: A temporal logic query checker. In *Proc. of the 15th International Conference on Computer Aided Verification (CAV)*, volume 2725 of *LNCS*, pages 210–214. Springer-Verlag, 2003.

[4] A. Gurfinkel, M. Chechik, and B. Devereux. Temporal logic query checking: A tool for model exploration. *IEEE Transactions on Software Engineering (TSE)*, 29(10):898–914, 2003.

[5] A. Gurfinkel, B. Devereux, and M. Chechik. Model exploration with temporal logic query checking. In *Proc. of the 10th ACM Symposium on Foundations of Software Engineering (FSE)*, pages 139–148. ACM, 2002.

[6] M. Samer. Temporal logic queries in model checking. Master's thesis, TU Vienna, May 2002.

[7] M. Samer. *Reasoning about Specifications in Model Checking*. PhD thesis, TU Vienna, Sept. 2004.

[8] M. Samer and H. Veith. Validity of CTL queries revisited. In *Proc. of the 12th Annual Conference of the European Association for Computer Science Logic (CSL)*, volume 2803 of *LNCS*, pages 470–483. Springer-Verlag, 2003.

[9] M. Samer and H. Veith. Parameterized vacuity. In *Proc. of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, volume 3312 of *LNCS*, pages 322–336. Springer-Verlag, 2004.

[10] M. Samer and H. Veith. A syntactic characterization of distributive LTL queries. In *Proc. of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *LNCS*, pages 1099–1110. Springer-Verlag, 2004.