A Resolution Calculus for the Branching-Time Temporal Logic CTL

LAN ZHANG, Capital University of Economics and Business, China
ULLRICH HUSTADT and CLARE DIXON, University of Liverpool, UK

The branching-time temporal logic CTL is useful for specifying systems that change over time and involve quantification over possible futures. Here we present a resolution calculus for CTL that involves the translation of formulae to a normal form and the application of a number of resolution rules. We use indices in the normal form to represent particular paths and the application of the resolution rules is restricted dependent on an ordering and selection function to reduce the search space. We show that the translation preserves satisfiability, the calculus is sound, complete, and terminating, and consider the complexity of the calculus.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Temporal logic; Mechanical theorem proving*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving

General Terms: Theory, Algorithms, Verification

Additional Key Words and Phrases: Temporal logic, automated theorem proving, resolution

ACM Reference Format:
Lan Zhang, Ullrich Hustadt, and Clare Dixon. 2014. A resolution calculus for the branching-time temporal logic CTL. ACM Trans. Comput. Logic 15, 1, Article 10 (February 2014), 38 pages.
DOI: http://dx.doi.org/10.1145/2529993

1. INTRODUCTION

Computation Tree Logic (CTL) [Clarke and Emerson 1982] is a propositional branching-time temporal logic whose underlying model of time is a choice of possibilities branching into the future. CTL uses the path quantifiers **A** (for all paths) and **E** (for some path) and temporal operators, for example, $\Box$ meaning now and at all future moments. Additionally, CTL has the restriction that each temporal operator must be paired with a path quantifier, for example, $\mathbf{A}\Box\varphi$ means on all paths φ always holds. There are many important applications that can be represented in and reasoned about in CTL, such as the verification of digital circuits [Clarke et al. 2000], analysis of real time and concurrent systems [Manna and Pnueli 1992], XPath query processing [Afanasiev et al. 2004], communication protocol verification [Clarke et al. 1986], and Grid Component system verification [Basso and Bolotov 2007]. A variant on CTL has been used in Attie [2003] applied to concurrent control protocols, where the focus is synthesising a concurrent program from a specification. Additionally,

This article is a revised and extended version of the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ for CTL described in Zhang et al. [2009a] and formed part of the thesis Zhang [2010].

This work was partially supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/D060451. L. Zhang was partially supported by the Technology Foundation for Selected Overseas Chinese Scholars, Bureau of Human Resources and Social Security of Beijing, by the National Natural Science Foundation of China (Grant No. 61303018), and by Research Improvement Funding, Beijing Municipal Education Commission.

Authors' addresses: L. Zhang, School of Information, Capital University of Economics and Business, China; U. Hustadt and C. Dixon (corresponding author), Department of Computer Science, University of Liverpool, U.K.; corresponding author's email: cldixon@liv.ac.uk.

1529-3785/2014/02-ART10
DOI: http://dx.doi.org/10.1145/2529993

branching-time logics have been used to specify and verify robustness in systems. The logics RoCTL* [French et al. 2007] and RoCTL [McCabe-Dansted and Dixon 2010] are branching-time temporal logics with additional operators to deal with obligation and robustness (i.e., at most n failures can occur).

Given the specification of a system in some logic it is often useful to be able to verify properties of that system. This is to show that desirable properties do hold on all runs through the system and unwanted properties do not hold. There are two main ways to do this: model checking and theorem proving. Using model checking one inputs a model of the system, usually a state transition system and a property to be checked that is a formula in some logical language, for example, CTL. The output is a set of states where the property holds. With theorem proving, both the model of the system and the property are given as logical formulae. A calculus for the logic is applied to show that the property follows from the specification. In this article, we follow the theorem proving route and provide a resolution-based calculus for CTL. The complexity of the model checking problem for CTL is PTIME complete [Clarke et al. 1986] and the complexity of satisfiability is EXPTIME-complete [Clarke and Emerson 1982; Emerson 1990; Emerson and Halpern 1985]. Whilst the complexity of satisfiability is higher than for model checking, theorem proving approaches to CTL are still useful, for example, where a model of a system is not readily available or immediately obvious, or to use in conjunction with model checkers for example to check that complex CTL properties are in fact satisfiable before they are model checked.

This article presents a sound, complete, and terminating resolution calculus, $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$, for the logic CTL. The overall approach involves transformation to a normal form, called Separated Normal Form with Global Clauses for CTL, $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ for short, and the application of *step* and *eventuality* resolution rules in $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ that deal with constraints on next states and on future states, respectively.

The calculus relates to that in Bolotov [2000] and we have improved upon it in the following aspects.

(1) An idea introduced in Bolotov [2000] is the use of indices as part of the CTL normal form that relate to a particular path. We give a formal interpretation of indices and a formal semantics for the indexed normal form, $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$, which is missing from Bolotov [2000].
(2) An ordering and a selection function are introduced into the calculus which allow us to reduce the number of possible applications of the inference rules during proof search.
(3) We show that our calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ is sound, complete, and terminating. Using our completeness proof, we can show that two eventuality resolution rules in Bolotov [2000] are redundant.
(4) A detailed complexity analysis of the calculus is provided, which is absent for the earlier calculus.
(5) Finally, we have implemented $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ in the theorem prover CTL-RP [Zhang et al. 2009b], whereas no implementation was provided for the earlier calculus in Bolotov [2000].

The rest of this article is organised as follows. We first present the syntax and semantics of CTL in Section 2 and then introduce a normal form for CTL, $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$, in Section 3. In Section 4 the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ is presented. We provide proofs for soundness and completeness of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ in Section 5. In Section 6, we discuss the complexity of our calculus $\mathsf{R}^{\succ,S}_{\text{CTL}}$. Finally, related work is discussed in Section 7, and conclusions are drawn in Section 8. Some of the proofs have been moved to the online Appendix.

2. COMPUTATIONAL TREE LOGIC (CTL)

The language of CTL is based on

—a set of atomic propositions P_{PL};
—propositional constants, **true** and **false**;
—boolean operators, $\wedge$, $\vee$, $\Rightarrow$, and $\neg$ ($\wedge$ and $\vee$ are associative and commutative[1]); and
—temporal operators $\square$ (always in the future), $\bigcirc$ (at the next moment in time), $\diamond$ (eventually in the future), $\mathcal{U}$ (until), and $\mathcal{W}$ (unless); and the *universal path quantifier* **A** (for all future paths) and the *existential path quantifier* **E** (for some future path).

The set of *(well-formed) formulae of CTL* is inductively defined as follows:

(1) **true** and **false** are CTL formulae;
(2) all atomic propositions in P_{PL} are CTL formulae; and
(3) if φ and ψ are CTL formulae, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, $\mathbf{A}\square\varphi$, $\mathbf{A}\diamond\varphi$, $\mathbf{A}\bigcirc\varphi$, $\mathbf{A}(\varphi\,\mathcal{U}\,\psi)$, $\mathbf{A}(\varphi\,\mathcal{W}\,\psi)$, $\mathbf{E}\square\varphi$, $\mathbf{E}\diamond\varphi$, $\mathbf{E}\bigcirc\varphi$, $\mathbf{E}(\varphi\,\mathcal{U}\,\psi)$, and $\mathbf{E}(\varphi\,\mathcal{W}\,\psi)$.

Formulae of CTL over P_{PL} are typically interpreted in *model structures* (see, e.g., [Emerson 1990]), $M = \langle S, R, L\rangle$, where S is a set of *states*; R is a serial binary *accessibility relation* over S; and $L : S \to 2^{\mathsf{P}_{\mathsf{PL}}}$ is an *interpretation function* mapping each state to the set of atomic propositions true at that state. These model structures are not required to be tree structures. However, CTL formulae can also be interpreted in tree model structures, which will be introduced later.

An infinite path χ_{s_i} is an infinite sequence of states $s_i, s_{i+1}, s_{i+2}, \ldots$ such that for every $j \geq i$, $(s_j, s_{j+1}) \in R$. A state $s' \in S$ is *reachable* from the state $s \in S$ iff there exists an infinite path χ_s such that $s' \in \chi_s$. If there exists two states $s, s' \in S$ such that $(s, s') \in R$, we say that s is a *predecessor* of s' and s' is a *successor* of s.

The satisfaction relation $\models$ between a pair consisting of a model structure M and a state $s_i \in S$, and a CTL formula is inductively defined as follows.

$$\begin{array}{lll}
\langle M, s_i\rangle \models \mathbf{true} & & \\
\langle M, s_i\rangle \not\models \mathbf{false} & & \\
\langle M, s_i\rangle \models p & \text{iff } p \in L(s_i) \text{ for an atomic proposition } p \in \mathsf{P}_{\mathsf{PL}} \\
\langle M, s_i\rangle \models \neg\varphi & \text{iff } \langle M, s_i\rangle \not\models \varphi \\
\langle M, s_i\rangle \models (\varphi \wedge \psi) & \text{iff } \langle M, s_i\rangle \models \varphi \text{ and } \langle M, s_i\rangle \models \psi \\
\langle M, s_i\rangle \models (\varphi \vee \psi) & \text{iff } \langle M, s_i\rangle \models \varphi \text{ or } \langle M, s_i\rangle \models \psi \\
\langle M, s_i\rangle \models (\varphi \Rightarrow \psi) & \text{iff } \langle M, s_i\rangle \not\models \varphi \text{ or } \langle M, s_i\rangle \models \psi \\
\langle M, s_i\rangle \models \mathbf{E}\bigcirc\psi & \text{iff there exists a path } \chi_{s_i} \text{ such that } \langle M, s_{i+1}\rangle \models \psi \\
\langle M, s_i\rangle \models \mathbf{A}(\varphi\,\mathcal{U}\,\psi) & \text{iff for every path } \chi_{s_i} \text{ there exists } s_j \in \chi_{s_i} \text{ such that} \\
 & \quad \langle M, s_j\rangle \models \psi \text{ and for every } s_k \in \chi_{s_i}, \text{ if } i \leq k < j, \\
 & \quad \text{then } \langle M, s_k\rangle \models \varphi \\
\langle M, s_i\rangle \models \mathbf{E}(\varphi\,\mathcal{U}\,\psi) & \text{iff there exists a path } \chi_{s_i} \text{ and there exists } s_j \in \chi_{s_i} \\
 & \quad \text{such that } \langle M, s_j\rangle \models \psi \text{ and for every } s_k \in \chi_{s_i}, \\
 & \quad \text{if } i \leq k < j, \text{ then } \langle M, s_k\rangle \models \varphi
\end{array}$$

[1]Used to simplify the presentation of the subsequent simplification and resolution rules.

In addition, we use the following equivalences to define the remaining operators of CTL.

$$\begin{array}{rlrl}
\mathbf{A}\Diamond\varphi \equiv & \mathbf{A}(\mathbf{true}\,\mathcal{U}\,\varphi) & \mathbf{E}\Diamond\varphi \equiv & \mathbf{E}(\mathbf{true}\,\mathcal{U}\,\varphi) \\
\mathbf{A}\Box\varphi \equiv & \neg\mathbf{E}\Diamond\neg\varphi & \mathbf{E}\Box\varphi \equiv & \neg\mathbf{A}\Diamond\neg\varphi \\
\mathbf{A}(\varphi\,\mathcal{W}\,\psi) \equiv & \neg\mathbf{E}(\neg\psi\,\mathcal{U}\,(\neg\varphi\wedge\neg\psi)) & \mathbf{E}(\varphi\,\mathcal{W}\,\psi) \equiv & \neg\mathbf{A}(\neg\psi\,\mathcal{U}\,(\neg\varphi\wedge\neg\psi)) \\
\mathbf{A}\bigcirc\varphi \equiv & \neg\mathbf{E}\bigcirc\neg\varphi & &
\end{array}$$

A CTL formula φ is *satisfiable*, iff for some model structure $M = \langle S, R, L\rangle$ and some state $s \in S$, $M, s \models \varphi$, and *unsatisfiable* otherwise. A model structure M such that φ is true at some state $s \in S$ is called a *model* of φ. A CTL formula φ is *valid*, written $\models \varphi$, iff for every model structure $M = \langle S, R, L\rangle$ and for every state $s \in S$, $M, s \models \varphi$. The satisfiability problem of CTL is known to be EXPTIME-complete [Clarke and Emerson 1982; Emerson 1990; Emerson and Halpern 1985].

The set of valid CTL formulae is not affected whether the model structures are tree model structures or not [Emerson 1990; Emerson and Halpern 1982]. Essentially, as R is a serial relation, any arbitrary model structure can be "unwound" into an infinite tree model structure. Therefore, in the following, we restrict ourselves to model structures $M = \langle S, R, L, s_0\rangle$ such that

—there exists a unique state s_0, called the *root*, such that every state $s \in S$ is reachable from state s_0 and there are no predecessors of s_0;
—for every state $s \in S$ except the root, state s has exactly one predecessor.

For model structures $M = \langle S, R, L, s_0\rangle$, it is also convenient to use definitions of satisfiability and validity which are slightly different from the version presented earlier. In particular, we say a CTL formula φ is satisfiable iff for some model structure $M = \langle S, R, L, s_0\rangle$, $M, s_0 \models \varphi$ and unsatisfiable otherwise. A model structure $M = \langle S, R, L, s_0\rangle$ such that φ is true at s_0 is a model of φ. A CTL formula φ is valid iff for every model structure $M = \langle S, R, L, s_0\rangle$, $M, s_0 \models \varphi$. Thus, our definition requires that if M is a model of φ, then φ must be satisfied at the root of M, namely, state s_0, whereas the previous definition allows φ to be satisfied at any state of M. It is not hard to see that there exists a model for a CTL formula φ according to the previous definition iff there exists a model for φ according to our definition. The reason we add this restriction is that it can simplify the proof that our transformation rules preserve satisfiability.

3. NORMAL FORM

The calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ operates on formulae in a clausal normal form, called Separated Normal Form with Global Clauses for CTL, denoted by $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$. An important difference between CTL formulae and $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ is an extension of the syntax of CTL to use *indices*. These are associated with the existential path quantifiers. Formulae such as $\mathbf{E}\bigcirc p$ (or $\mathbf{E}\Diamond p$) are decorated with an index *ind* to become $\mathbf{E}_{\langle ind\rangle}\bigcirc p$ (respectively, $\mathbf{E}_{\langle ind\rangle}\Diamond p$) to denote that p is satisfied on a particular successor state (respectively, state accessible by a particular path). They are used to preserve satisfiability during the transformation into the normal form. The language of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses is defined over an extension of CTL. That is the language is based on

—the language of CTL;
—a propositional constant **start**; and
—a countably infinite index set Ind.

To improve the readability of clauses, we introduce an operator precedence which allows us to reduce the number of parentheses required. We associate each operator with one of the following five precedence groups, where (i) is highest and (v) is lowest.

(i) $\mathbf{A}\bigcirc, \mathbf{E}\bigcirc, \mathbf{A}\Diamond, \mathbf{E}\Diamond, \mathbf{A}\Box, \mathbf{E}\Box, \mathbf{A}\mathcal{U}, \mathbf{E}\mathcal{U}, \mathbf{A}\mathcal{W}, \mathbf{E}\mathcal{W}, \mathbf{E}\bigcirc_{\langle ind\rangle}, \mathbf{E}_{\langle ind\rangle}\Diamond, \mathbf{E}_{\langle ind\rangle}\Box,$ $\mathbf{E}_{\langle ind\rangle}\mathcal{U}, \mathbf{E}_{\langle ind\rangle}\mathcal{W}$, where $ind \in \mathsf{Ind}$
(ii) $\neg$
(iii) $\wedge$
(iv) $\vee$
(v) $\Rightarrow$

Two operators in the same group have the same precedence. Higher precedence operators are applied before lower precedence operators. Then the language of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses* consists of formulae of the following forms:

$$\begin{array}{ll}
\mathbf{A}\Box(\mathbf{start} \Rightarrow \bigvee_{j=1}^{k} m_j) & \text{(initial clause)} \\
\mathbf{A}\Box(\mathbf{true} \Rightarrow \bigvee_{j=1}^{k} m_j) & \text{(global clause)} \\
\mathbf{A}\Box(\bigwedge_{i=1}^{n} l_i \Rightarrow \mathbf{A}\bigcirc \bigvee_{j=1}^{k} m_j) & \text{(}\mathbf{A}\text{-step clause)} \\
\mathbf{A}\Box(\bigwedge_{i=1}^{n} l_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc \bigvee_{j=1}^{k} m_j) & \text{(}\mathbf{E}\text{-step clause)} \\
\mathbf{A}\Box(\bigwedge_{i=1}^{n} l_i \Rightarrow \mathbf{A}\Diamond l) & \text{(}\mathbf{A}\text{-sometime clause)} \\
\mathbf{A}\Box(\bigwedge_{i=1}^{n} l_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond l) & \text{(}\mathbf{E}\text{-sometime clause),}
\end{array}$$

where $k \geq 0$, $n > 0$, **start** is a propositional constant, l_i $(1 \leq i \leq n)$, m_j $(1 \leq j \leq k)$, and l are literals, that is, atomic propositions or their negation, and *ind* is an element of Ind. For a literal l, $\neg l$ denotes p if l is $\neg p$ and it denotes $\neg p$ if l is p, for some atomic proposition p. We use '**false**' to denote both the logical constant and empty disjunctions in clauses. As all clauses are of the form $\mathbf{A}\Box(P \Rightarrow D)$, we often simply write $P \Rightarrow D$ instead. We assume that all $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses are always kept in condensed form, that is, there are no duplicate literals in either P or D. For example, a $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clause $r \wedge q \wedge q \Rightarrow \mathbf{A}\bigcirc(q \vee p \vee q)$ is always represented as $r \wedge q \Rightarrow \mathbf{A}\bigcirc(q \vee p)$. We call a clause which is either an initial, a global, an $\mathbf{A}$-step, or an $\mathbf{E}$-step clause a *determinate clause*. The formula $\mathbf{A}\Diamond l$ is called an ***A**-eventuality* and the formula $\mathbf{E}_{\langle ind\rangle}\Diamond l$ is called an ***E**-eventuality*.

A key part of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ are indices. These indices can be used to preserve a particular path context. For example,

—the formula $\mathbf{E}\bigcirc p \wedge \mathbf{E}\bigcirc\neg p$ is obviously satisfiable, as in general, these two existential path quantifiers can refer to two different paths; whereas
—the formula $\mathbf{E}_{\langle ind\rangle}\bigcirc p \wedge \mathbf{E}_{\langle ind\rangle}\bigcirc\neg p$ is unsatisfiable, as two existential quantifiers having the same index *ind* indicate p and $\neg p$ are satisfied at the same successor state of the current state.

Informally speaking, when an index *ind* is associated with a next operator ($\bigcirc$), this allows the identification of a particular successor state. For instance, if $\mathbf{E}_{\langle ind\rangle}\bigcirc p$ is satisfied at a state s, then there exists a state s' such that the edge from s to s' is labelled by *ind* and p holds at s'. When an index *ind* is associated with a multistep operator ($\Diamond$, $\Box$, $\mathcal{U}$ or $\mathcal{W}$), *ind* allows the identification of a particular path, where the edge to each successor state in the path is is labelled by *ind*. For instance, if $\mathbf{E}_{\langle ind\rangle}\Diamond p$ is satisfied at state s, then there exists a state s' reachable from s such that each edge in the path from s to s' is labelled by *ind* and p holds at s'.

We use indices in the transformation to normal form to preserve satisfiability. In particular, they are necessary to translate until formulae (e.g., $\mathbf{E}(p\mathcal{U}l)$). Here the transformation (see Section 3.2, *Trans*(11)) introduces global and $\mathbf{E}$-step clauses that enforce $l \vee p$ in the next moment on some path. However, we must ensure that l holds eventually by introducing a $\mathbf{E}$-sometime clause with $\mathbf{E}\Diamond l$ on the right-hand side. This must hold on the *same path* as the **E**-step clauses mentioned previously, and we use indices to ensure this.

3.1. Syntax and Semantics of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$

To provide a semantics for $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$, we extend model structures $\langle S, R, L, s_0\rangle$ to $\langle S, R, L, [\_], s_0\rangle$, where the unary function $[\_] : \mathsf{Ind} \to 2^{(S\times S)}$ maps every index $ind \in \mathsf{Ind}$ to a *successor function* $[ind]$ which is a total functional relation on S and a subset of the binary accessibility relation R, that is, for every $s \in S$, there exists exactly one state $s' \in S$ such that $(s, s') \in [ind]$ and $(s, s') \in R$. So $[\_]$ applied to *ind* results in a set of pairs denoting, for any state, exactly one *ind*-related successor state. A state $s' \in S$ is an *ind-successor state* of state $s \in S$ iff $(s, s') \in [ind]$. An *infinite path* $\chi^{\langle ind\rangle}_{s_i}$ is an infinite sequence of states $s_i, s_{i+1}, s_{i+2}, \ldots$ such that for every $j \geq i$, $(s_j, s_{j+1}) \in [ind]$. An infinite path $\chi^{\langle ind\rangle}_{s_k} = s'_k, s'_{k+1}, \ldots$ is a *subpath* of $\chi^{\langle ind\rangle}_{s_i}$ iff there exists a natural number $l, l \geq i$, where $s_l \in \chi^{\langle ind\rangle}_{s_i}$, such that for every $j \geq 0$, $s'_{k+j} = s_{l+j}$. Note that since $[ind]$ is a function, for every state $s_i \in S$, there exists exactly one infinite path $\chi^{\langle ind\rangle}_{s_i}$ and for every state $s_j \in \chi^{\langle ind\rangle}_{s_i}$, $\chi^{\langle ind\rangle}_{s_j}$ is a subpath of $\chi^{\langle ind\rangle}_{s_i}$. The semantics of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ is then defined as shown next as an extension of the semantics of CTL defined in Section 2. Although the operators $\mathbf{E}_{\langle ind\rangle}\Box$, $\mathbf{E}_{\langle ind\rangle}\,\mathcal{U}$ and $\mathbf{E}_{\langle ind\rangle}\,\mathcal{W}$ do not appear in the normal form, we state their semantics, because they occur in the normal form transformation. (The semantics of the remaining operators is analogous to that given previously but in the extended model structure $\langle S, R, L, [\_], s_0\rangle$.)

$$\begin{array}{ll}
\langle M, s_i\rangle \models \mathbf{start} & \text{iff } s_i = s_0\\
\langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}\bigcirc\psi & \text{iff for the path } \chi^{\langle ind\rangle}_{s_i},\ \langle M, s_{i+1}\rangle \models \psi\\
\langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}\Diamond\psi & \text{iff } \langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}(\mathbf{true}\,\mathcal{U}\,\psi)\\
\langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}\Box\psi & \text{iff for every } s_j \in \chi^{\langle ind\rangle}_{s_i},\ \langle M, s_j\rangle \models \psi\\
\langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}(\varphi\,\mathcal{U}\,\psi) & \text{iff there exists } s_j \in \chi^{\langle ind\rangle}_{s_i} \text{ such that } \langle M, s_j\rangle \models \psi \text{ and}\\
 & \quad\text{for every } s_k \in \chi^{\langle ind\rangle}_{s_i},\ \text{if } i \leq k < j, \text{ then } \langle M, s_k\rangle \models \varphi\\
\langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}(\varphi\,\mathcal{W}\,\psi) & \text{iff } \langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}\Box\varphi \text{ or } \langle M, s_i\rangle \models \mathbf{E}_{\langle ind\rangle}(\varphi\,\mathcal{U}\,\psi)
\end{array}$$

A $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ formula φ is *satisfiable*, iff for some model structure $M{=}\langle S, R, L, [\_], s_0\rangle, M, s_0 \models \varphi$, and *unsatisfiable* otherwise. A model structure $M{=}\ \langle S, R, L, [\_], s_0\rangle$ such that φ is true at the state $s_0 \in S$ is called a *model* of φ, and we say that M satisfies φ. A $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ formula φ is *valid*, written $\models \varphi$, iff for every model structure $M = \langle S, R, L, [\_], s_0\rangle, M, s_0 \models \varphi$.

3.2. Transformation

We first introduce definitions of *indexed CTL formula* and *CTL clauses*, which will be used in our definition of the transformation from an arbitrary CTL formula into a set of formulae in normal form.

Definition 3.1 (*Indexed CTL Formula*). The set of *indexed CTL formulae* is inductively defined as follows:

(1) **true**, **false** and **start** are indexed CTL formulae;
(2) all atomic propositions in P_{PL} are indexed CTL formulae; and
(3) if φ and ψ are indexed CTL formulae, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, $\mathbf{A}\Box\varphi$, $\mathbf{A}\Diamond\varphi$, $\mathbf{A}\bigcirc\varphi$, $\mathbf{A}(\varphi\,\mathcal{U}\,\psi)$, $\mathbf{A}(\varphi\,\mathcal{W}\,\psi)$, $\mathbf{E}\Box\varphi$, $\mathbf{E}\Diamond\varphi$, $\mathbf{E}\bigcirc\varphi$, $\mathbf{E}(\varphi\,\mathcal{U}\,\psi)$, $\mathbf{E}(\varphi\,\mathcal{W}\,\psi)$, $\mathbf{E}_{\langle ind\rangle}\Box\varphi$, $\mathbf{E}_{\langle ind\rangle}\Diamond\varphi$, $\mathbf{E}_{\langle ind\rangle}\bigcirc\varphi$, $\mathbf{E}_{\langle ind\rangle}(\varphi\,\mathcal{U}\,\psi)$, and $\mathbf{E}_{\langle ind\rangle}(\varphi\,\mathcal{W}\,\psi)$, where *ind* is an arbitrary index in Ind.

Definition 3.2 (*CTL Clauses*). A CTL formula of the form $\mathbf{A}\Box(P \Rightarrow \varphi)$, where P is a conjunction of literals (possibly consisting of a single literal) or a propositional constant and φ is an arbitrary indexed CTL formula, is a *CTL clause* or a *clause*.

We now define a set of transformation rules which allows us to transform an arbitrary CTL formula into an equi-satisfiable set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses.

Let *nnf* denote a function which transforms an arbitrary CTL formula into its negation normal form by pushing negations 'inwards'. Let *simp* be a function which simplifies an arbitrary CTL formula by exhaustive application of the following simplification rules:

$$\begin{array}{rr} (\varphi \wedge \mathbf{true}) \longrightarrow \varphi & (\varphi \wedge \mathbf{false}) \longrightarrow \mathbf{false} \\ (\varphi \vee \mathbf{true}) \longrightarrow \mathbf{true} & (\varphi \vee \mathbf{false}) \longrightarrow \varphi \\ \neg\mathbf{true} \longrightarrow \mathbf{false} & \neg\mathbf{false} \longrightarrow \mathbf{true}, \end{array}$$

where φ is a CTL formula and $\vee$ and $\wedge$ are commutative and associative, plus the following rules which are based on the equivalences in Emerson [1990]:

$$\begin{array}{rr} \mathbf{PTfalse} \longrightarrow \mathbf{false} & \mathbf{PTtrue} \longrightarrow \mathbf{true} \\ \mathbf{P}(\varphi\,\mathcal{U}\,\mathbf{false}) \longrightarrow \mathbf{false} & \mathbf{P}(\varphi\,\mathcal{U}\,\mathbf{true}) \longrightarrow \mathbf{true} \\ \mathbf{P}(\mathbf{false}\,\mathcal{U}\,\varphi) \longrightarrow \varphi & \mathbf{P}(\mathbf{true}\,\mathcal{U}\,\varphi) \longrightarrow \mathbf{P}\Diamond\varphi \\ \mathbf{P}(\varphi\,\mathcal{W}\,\mathbf{false}) \longrightarrow \mathbf{P}\Box\varphi & \mathbf{P}(\varphi\,\mathcal{W}\,\mathbf{true}) \longrightarrow \mathbf{true} \\ \mathbf{P}(\mathbf{false}\,\mathcal{W}\,\varphi) \longrightarrow \varphi & \mathbf{P}(\mathbf{true}\,\mathcal{W}\,\varphi) \longrightarrow \mathbf{true}, \end{array}$$

where $\mathbf{P} \in \{\mathbf{A}, \mathbf{E}\}$ and $\mathbf{T} \in \{\bigcirc, \Box, \Diamond\}$.

Let *init*(φ) be the set of CTL clauses $\{\mathbf{A}\Box(\mathbf{start} \Rightarrow p), \mathbf{A}\Box(p \Rightarrow simp(nnf(\varphi)))\}$, where p is a new atomic proposition in P_{PL} that does not occur in φ.

Then the transformation of an arbitrary CTL formula φ into $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ consists of a sequence $T_0, T_1, \ldots, T_n$ of sets of CTL clauses such that (i) $T_0 = init(\varphi)$ and (ii) for every t, $0 \leq t < n$, $T_{t+1} = (T_t \setminus \{\psi\}) \cup R_t$, where ψ is a formula in T_t not in $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ and R_t is the result of applying a matching transformation rule to ψ. Moreover, for every t, $0 \leq t < n$, T_t contains at least one formula not in $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ while all formulae in T_n are in $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$.

Note that for each rule of *Trans* containing a proposition p, p represents a new atomic proposition in P_{PL} which does not occur in T_t when we apply the rule to a clause in T_t. Furthermore, in the presentation of the rules, let

—q be an atomic proposition,
—l be a literal,
—D be a disjunction of literals (possible consisting of a single literal), and
—φ, φ_1, and φ_2 be CTL formulae.

The definition of the rule set *Trans* is as follows.

—Index introduction rules:

Trans(1) $\quad q \Rightarrow \mathbf{ET}\varphi \longrightarrow q \Rightarrow \mathbf{E}_{\langle ind\rangle}\mathbf{T}\varphi$

Trans(2) $\quad q \Rightarrow \mathbf{E}(\varphi_1\mathbf{T}'\varphi_2) \longrightarrow q \Rightarrow \mathbf{E}_{\langle ind\rangle}(\varphi_1\mathbf{T}'\varphi_2)$

where $\mathbf{T} \in \{\bigcirc, \Box, \Diamond\}$, $\mathbf{T}' \in \{\mathcal{U}, \mathcal{W}\}$ and *ind* is a new index.

—Boolean rules:

$$\textit{Trans}(3) \qquad q \Rightarrow \varphi_1 \wedge \varphi_2 \longrightarrow \begin{cases} q \Rightarrow \varphi_1 \\ q \Rightarrow \varphi_2 \end{cases}$$

$$\textit{Trans}(4) \qquad q \Rightarrow \varphi_1 \vee \varphi_2 \longrightarrow \begin{cases} q \Rightarrow \varphi_1 \vee p, & \text{if } \varphi_2 \text{ is not a disjunction} \\ p \Rightarrow \varphi_2, & \text{of literals.} \end{cases}$$

$$\textit{Trans}(5) \qquad q \Rightarrow D \longrightarrow \textbf{true} \Rightarrow \neg q \vee D$$

$$q \Rightarrow \textbf{false} \longrightarrow \textbf{true} \Rightarrow \neg q$$

$$q \Rightarrow \textbf{true} \longrightarrow \{\}$$

—Temporal operator rules:

$$\textit{Trans}(6) \qquad q \Rightarrow \mathbf{P}\bigcirc\varphi \longrightarrow \begin{cases} q \Rightarrow \mathbf{P}\bigcirc p & \text{if } \varphi \text{ is not a disjunction} \\ p \Rightarrow \varphi & \text{of literals.} \end{cases}$$

$$\textit{Trans}(7) \qquad q \Rightarrow \mathbf{P}\Diamond\varphi \longrightarrow \begin{cases} q \Rightarrow \mathbf{P}\Diamond p \\ p \Rightarrow \varphi \end{cases} \quad \text{if } \varphi \text{ is not a literal.}$$

$$\textit{Trans}(8) \qquad q \Rightarrow \mathbf{P}(\varphi_1 \mathcal{U} \varphi_2) \longrightarrow \begin{cases} q \Rightarrow \mathbf{P}(\varphi_1 \mathcal{U}\, p) \\ p \Rightarrow \varphi_2 \end{cases} \quad \text{if } \varphi_2 \text{ is not a literal.}$$

$$\textit{Trans}(9) \qquad q \Rightarrow \mathbf{P}(\varphi_1 \mathcal{W} \varphi_2) \longrightarrow \begin{cases} q \Rightarrow \mathbf{P}(\varphi_1 \mathcal{W}\, p) \\ p \Rightarrow \varphi_2 \end{cases} \quad \text{if } \varphi_2 \text{ is not a literal.}$$

$$\textit{Trans}(10) \qquad q \Rightarrow \mathbf{P}\Box\varphi \longrightarrow \begin{cases} q \Rightarrow p \\ p \Rightarrow \varphi \\ p \Rightarrow \mathbf{P}\bigcirc p \end{cases}$$

$$\textit{Trans}(11) \qquad q \Rightarrow \mathbf{P}(\varphi \mathcal{U}\, l) \longrightarrow \begin{cases} q \Rightarrow l \vee p \\ p \Rightarrow \varphi \\ p \Rightarrow \mathbf{P}\bigcirc(l \vee p) \\ q \Rightarrow \mathbf{P}\Diamond l \end{cases}$$

$$\textit{Trans}(12) \qquad q \Rightarrow \mathbf{P}(\varphi \mathcal{W}\, l) \longrightarrow \begin{cases} q \Rightarrow l \vee p \\ p \Rightarrow \varphi \\ p \Rightarrow \mathbf{P}\bigcirc(l \vee p) \end{cases}$$

where $\mathbf{P} \in \{\mathbf{A}, \mathbf{E}_{\langle ind \rangle}\}$.

Example 3.3. In the following, we transform the unsatisfiable CTL formula $\varphi_2 = \mathbf{E}\Box\neg l \wedge \mathbf{A}\Diamond l$ into a satisfiability equivalent set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses. We apply the function *init* to the CTL formula φ_2:

$$\Gamma_2 = \textit{init}(\varphi_2) = \{\mathbf{A}\Box(\textbf{start} \Rightarrow p_1), \mathbf{A}\Box(p_1 \Rightarrow \mathbf{E}\Box\neg l \wedge \mathbf{A}\Diamond l)\}.$$

Then we transform the set Γ_2 of clauses into a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses.

1.	$\mathbf{start} \Rightarrow p_1$	
2.	$p_1 \Rightarrow \mathbf{E}\Box\neg l \wedge \mathbf{A}\Diamond l$	
3.	$p_1 \Rightarrow \mathbf{E}\Box\neg l$	$[2, \mathit{Trans}(3)]$
4.	$p_1 \Rightarrow \mathbf{A}\Diamond l$	$[2, \mathit{Trans}(3)]$
5.	$p_1 \Rightarrow \mathbf{E}_{\langle 1\rangle}\Box\neg l$	$[3, \mathit{Trans}(1)]$
6.	$p_1 \Rightarrow p_2$	$[5, \mathit{Trans}(10)]$
7.	$p_2 \Rightarrow \neg l$	$[5, \mathit{Trans}(10)]$
8.	$p_2 \Rightarrow \mathbf{E}_{\langle 1\rangle}\bigcirc p_2$	$[5, \mathit{Trans}(10)]$
9.	$\mathbf{true} \Rightarrow \neg p_1 \vee p_2$	$[6, \mathit{Trans}(5)]$
10.	$\mathbf{true} \Rightarrow \neg p_2 \vee \neg l$	$[7, \mathit{Trans}(5)]$

Then the set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses consisting of the following clauses is satisfiable iff the CTL formula $\mathbf{E}\Box\neg l \wedge \mathbf{A}\Diamond l$ is satisfiable.

1.	$\mathbf{start} \Rightarrow p_1$
4.	$p_1 \Rightarrow \mathbf{A}\Diamond l$
8.	$p_2 \Rightarrow \mathbf{E}_{\langle 1\rangle}\bigcirc p_2$
9.	$\mathbf{true} \Rightarrow \neg p_1 \vee p_2$
10.	$\mathbf{true} \Rightarrow \neg p_2 \vee \neg l$

We will return to this set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ later.

4. THE CLAUSAL RESOLUTION CALCULUS $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$

Our clausal resolution calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ for CTL is based on, but not identical to, the resolution calculus in Bolotov [2000] and Bolotov and Fisher [1999]. The calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ consists of

—eight *step* resolution rules SRES1 to SRES8,
—two *eventuality* resolution rules ERES1 and ERES2, and
—two *rewrite* rules RW1 and RW2.

Furthermore, all the rules of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ operate on $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses. The calculus can be used to develop an EXPTIME decision procedure for the satisfiability problem of CTL.

4.1. Step Resolution

As the search space for resolution for classical propositional and first-order logic is large, in practice, refinements for resolution are necessary. In particular, ordering and selection function refinements are utilised by many efficient theorem provers, for example, SPASS,[2] Vampire,[3] and Prover9.[4] The need for such refinements in order to be efficient in practice is also true for non-classical logics, for instance, CTL. Moreover, $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ is designed in such a way that the step resolution can be emulated by first-order resolution. Motivated by refinements of propositional and first-order resolution [Bachmair and Ganzinger 2001], we restrict the applicability of step resolution rules by means of an atom ordering and a selection function. These refinements have two advantages: (i) we can prove that they do not impair the completeness of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$; and (ii) the efforts of implementing them can be dramatically reduced by reusing some existing high performance first-order resolution prover, which contains these refinements.

[2]`http://www.spass-prover.org/download/.`
[3]`http://vprover.org/index.cgi.`
[4]Prover9 and Mace 4. `http://www.cs.unm.edu/~mccune/prover9/.`

Before we introduce the atom ordering and the selection function we use for step resolution, we first give the following definitions.

Definition 4.1 (*Partial Ordering*). A *partial ordering* R on a set S is the ordering such that the following hold.

—For every element $s \in S$, $(s, s) \notin R$.
—For all elements s, t, u of S, if $(s, t), (t, u) \in R$, then $(s, u) \in R$.

Definition 4.2 (*Total Ordering*). A partial ordering R on a set S is a *total ordering* if for every pair of distinct elements s and t of S, $(s, t) \in R$ or $(t, s) \in R$.

Definition 4.3 (*Well-Founded Ordering*). A partial ordering R on a set S is a *well-founded ordering* if every non-empty subset of S has a minimal element with respect to R.

An *atom ordering* for $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ is a well-founded and total ordering $\succ$ on the set P_{PL}. The ordering $\succ$ is extended to literals such that for each $p \in \mathsf{P}_{\mathsf{PL}}$, $\neg p \succ p$, and for each $q \in \mathsf{P}_{\mathsf{PL}}$ such that $q \succ p$ then $q \succ \neg p$ and $\neg q \succ \neg p$.

A literal l is (*strictly*) *maximal* with respect to a propositional disjunction C iff for every literal l' in C, $l' \not\succ l$ ($l' \not\succeq l$).

A *selection function* is an arbitrary function S mapping every propositional disjunction C to a possibly empty subset $S(C)$ of the negative literals occurring in C. If $l \in S(C)$ for a disjunction C, then we say that l is *selected* in C.

In the following presentation of the rules of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$, *ind* is an index in Ind, P and Q are conjunctions of literals, C and D are disjunctions of literals, neither of which contain duplicate literals, and l is a literal.

SRES1

$$\frac{\begin{array}{r} P \Rightarrow \mathbf{A}\bigcirc(C \vee l) \\ Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l) \end{array}}{P \wedge Q \Rightarrow \mathbf{A}\bigcirc(C \vee D)}$$

SRES2

$$\frac{\begin{array}{r} P \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(C \vee l) \\ Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l) \end{array}}{P \wedge Q \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(C \vee D)}$$

SRES3

$$\frac{\begin{array}{r} P \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(C \vee l) \\ Q \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(D \vee \neg l) \end{array}}{P \wedge Q \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(C \vee D)}$$

SRES4

$$\frac{\begin{array}{l} \mathbf{start} \Rightarrow C \vee l \\ \mathbf{start} \Rightarrow D \vee \neg l \end{array}}{\mathbf{start} \Rightarrow C \vee D}$$

SRES5

$$\frac{\begin{array}{r} \mathbf{true} \Rightarrow C \vee l \\ \mathbf{start} \Rightarrow D \vee \neg l \end{array}}{\mathbf{start} \Rightarrow C \vee D}$$

SRES6

$$\frac{\begin{array}{r} \mathbf{true} \Rightarrow C \vee l \\ Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l) \end{array}}{Q \Rightarrow \mathbf{A}\bigcirc(C \vee D)}$$

SRES7

$$\frac{\begin{array}{r} \mathbf{true} \Rightarrow C \vee l \\ Q \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(D \vee \neg l) \end{array}}{Q \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(C \vee D)}$$

SRES8

$$\frac{\begin{array}{l} \mathbf{true} \Rightarrow C \vee l \\ \mathbf{true} \Rightarrow D \vee \neg l \end{array}}{\mathbf{true} \Rightarrow C \vee D}$$

A step resolution rule, SRES1 to SRES8, is only applicable if and only if one of the following two conditions is satisfied:

(C1) if l is a positive literal, then

(1) l must be strictly maximal with respect to C and no literal is selected in $C \vee l$, and
(2) (i) $\neg l$ must be selected in $D \vee \neg l$ or (ii) no literal is selected in $D \vee \neg l$ and $\neg l$ is maximal with respect to D; or

(C2) if l is a negative literal, then

(1) (i) l must be selected in $C \vee l$ or (ii) no literal is selected in $C \vee l$ and l is maximal with respect to C, and
(2) $\neg l$ must be strictly maximal with respect to D and no literal is selected in $D \vee \neg l$.

Note that these two conditions are identical modulo the polarity of l, that is, having or not having the negation $\neg$ in front of l. If l in $C \vee l$ and $\neg l$ in $D \vee \neg l$ satisfy condition (C1) or condition (C2), then we say that l is *eligible* in $C \vee l$ and $\neg l$ is *eligible* in $D \vee \neg l$.

The rewrite rules RW1 and RW2 are defined as follows:

$$\textbf{RW1} \qquad \bigwedge_{i=1}^{n} m_i \Rightarrow \mathbf{A}\bigcirc\mathbf{false} \longrightarrow \mathbf{true} \Rightarrow \bigvee_{i=1}^{n} \neg m_i,$$

$$\textbf{RW2} \qquad \bigwedge_{i=1}^{n} m_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\mathbf{false} \longrightarrow \mathbf{true} \Rightarrow \bigvee_{i=1}^{n} \neg m_i,$$

where $n \geq 1$ and each m_i, $1 \leq i \leq n$, is a literal.

An example of how to apply step resolution and rewrite rules is given next.

Example 4.4. In Example 3.3, we have seen that the application of our transformation rules to the CTL formula $\varphi = \mathbf{E}\Box\neg l \wedge \mathbf{A}\Diamond l$ results in the following satisfiability equivalent set of $\text{SNF}^{\text{g}}_{\text{CTL}}$ clauses. Using the ordering $l \succ p_1 \succ p_2$ and the selection function S mapping every propositional disjunction C to an empty set, the underlined literals are the eligible literals in these clauses.

$$\begin{array}{rl}
1. & \mathbf{start} \Rightarrow \underline{p_1} \\
2. & p_1 \Rightarrow \mathbf{A}\Diamond\underline{l} \\
3. & p_2 \Rightarrow \mathbf{E}_{\langle 1\rangle}\bigcirc\underline{p_2} \\
4. & \mathbf{true} \Rightarrow \underline{\neg p_1} \vee p_2 \\
5. & \mathbf{true} \Rightarrow \neg p_2 \vee \underline{\neg l}
\end{array}$$

Using step resolution, we are able to generate the following derivation, where the notation $[c_1, \ldots, c_n, \text{SRESi}]$ indicates that we apply the ith step resolution rule to clauses $c_1, \ldots, c_n$, respectively.

$$6.\ \mathbf{start} \Rightarrow \underline{p_2} \qquad [1, 4, \text{SRES5}]$$

Note that without ordering and selection, we would also be able to derive the following two clauses.

$$\begin{array}{rll}
7. & p_2 \Rightarrow \mathbf{E}_{\langle 1\rangle}\bigcirc\neg l & [3, 5, \text{SRES7}] \\
8. & \mathbf{true} \Rightarrow \neg p_2 \vee \neg l & [4, 5, \text{SRES8}]
\end{array}$$

The importance of the ordering and selection function is that they restrict the applicability of the resolution rules. Example 4.4 illustrates that additional clauses may be derived without this. Although this example has an empty selection function, we can use a non-empty selection function to 'simulate' other well-known refinements of resolution. For example, we can 'simulate' positive resolution [Robinson 1965] using a selection function which always selects some negative literal if a clause has one. The completeness of this refinement of temporal resolution for the linear-time temporal logic PLTL was independently shown by Fernández Gago [2004].

4.2. Eventuality Resolution

The intuition underlying the following eventuality resolution rule ERES1 is to resolve an eventuality $\mathbf{A}\Diamond\neg l$, which states that $\Diamond\neg l$ is true on all paths, with a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, which, together, provided that their combined left-hand sides were satisfied, imply that $\Box l$ holds on (at least) one path.

ERES1

$$\frac{\begin{array}{c} P^{\dagger} \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box l \\ Q \Rightarrow \mathbf{A}\Diamond\neg l \end{array}}{Q \Rightarrow \mathbf{A}(\neg(P^{\dagger})\,\mathcal{W}\,\neg l)},$$

where $P^{\dagger} \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box l$ represents a set, $\Lambda_{\mathbf{E}\Box}$, of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses

$$\begin{array}{ccc} P^1_1 \Rightarrow *C^1_1 & & P^n_1 \Rightarrow *C^n_1 \\ \vdots & & \vdots \\ P^1_{m_1} \Rightarrow *C^1_{m_1} & \cdots & P^n_{m_n} \Rightarrow *C^n_{m_n}, \end{array}$$

where for every i, $1 \leq i \leq n$, there is some $ind \in \mathsf{Ind}$ such that each $*$ is either empty or an operator in $\{\mathbf{A}\bigcirc, \mathbf{E}_{\langle ind\rangle}\bigcirc\}$ and for every i, $1 \leq i \leq n$,

$$\left(\bigwedge_{j=1}^{m_i} C^i_j\right) \Rightarrow l, \qquad (1)$$

and

$$\left(\bigwedge_{j=1}^{m_i} C^i_j\right) \Rightarrow \left(\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} P^i_j\right), \qquad (2)$$

are provable. Furthermore, $P^{\dagger} = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} P^i_j$. Conditions (1) and (2) ensure that the set $\Lambda_{\mathbf{E}\Box}$ of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses implies $P^{\dagger} \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box l$.

Note that the conclusion of ERES1 is not stated in normal form. To present the conclusion of ERES1 in normal form, we use a new atomic proposition $w^{\mathbf{A}}_{\neg l}$ uniquely associated with the eventuality $\mathbf{A}\Diamond\neg l$. Then the conclusion of ERES1 can be represented by the following set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses.

$$\begin{array}{l} \left\{w^{\mathbf{A}}_{\neg l} \Rightarrow \mathbf{A}\bigcirc(\neg l \vee \bigvee_{j=1}^{m_i} \neg P^i_j) \mid 1 \leq i \leq n\right\} \\ \cup \left\{\mathbf{true} \Rightarrow \neg Q \vee \neg l \vee \bigvee_{j=1}^{m_i} \neg P^i_j \mid 1 \leq i \leq n\right\} \\ \cup \left\{\mathbf{true} \Rightarrow \neg Q \vee \neg l \vee w^{\mathbf{A}}_{\neg l},\ w^{\mathbf{A}}_{\neg l} \Rightarrow \mathbf{A}\bigcirc(\neg l \vee w^{\mathbf{A}}_{\neg l})\right\}. \end{array}$$

The use of a proposition $w^{\mathbf{A}}_{\neg l}$ uniquely associated with the eventuality $\mathbf{A}\Diamond\neg l$ is important for the termination of our procedure. It allows us to represent all resolvents by ERES1 using a fixed set of propositions depending only on the initial set of clauses, that is, n different $\mathbf{A}$-eventualities in the initial set of clauses require at most n new atomic propositions to represent resolvents by ERES1.

In the following, we give a concrete example to demonstrate an application of ERES1.

Example 4.5. We resolve the $\mathbf{A}$-sometime clause $u \Rightarrow \mathbf{A}\Diamond\neg l$ with the following set $\Lambda_{\mathbf{E}\Box}$ of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses.

$$\begin{array}{ll} p \Rightarrow \mathbf{E}_{\langle 1\rangle}\bigcirc l & q \Rightarrow \mathbf{E}_{\langle 2\rangle}\bigcirc l \\ r \Rightarrow \mathbf{E}_{\langle 1\rangle}\bigcirc q & q \Rightarrow \mathbf{E}_{\langle 2\rangle}\bigcirc p \\ & q \Rightarrow \mathbf{E}_{\langle 2\rangle}\bigcirc r, \end{array}$$

where each column satisfies (1) and (2). Therefore, $P^\dagger = ((p \wedge r) \vee q)$ and $P^\dagger \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box l$. We then can resolve the clause $u \Rightarrow \mathbf{A}\Diamond\neg l$ with $((p \wedge r) \vee q) \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box l$ as follows.

$$\frac{\begin{array}{c}((p \wedge r) \vee q) \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box l \\ u \Rightarrow \mathbf{A}\Diamond\neg l\end{array}}{u \Rightarrow \mathbf{A}(\neg((p \wedge r) \vee q)\,\mathcal{W}\,\neg l)}.$$

Then the resolvents of the application of ERES1 in $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ are as follows:

$$\begin{array}{ll}
w^{\mathbf{A}}_{\neg l} & \Rightarrow \mathbf{A}\bigcirc(\neg l \vee \neg p \vee \neg r) \\
w^{\mathbf{A}}_{\neg l} & \Rightarrow \mathbf{A}\bigcirc(\neg l \vee \neg q) \\
\mathbf{true} & \Rightarrow \neg u \vee \neg l \vee \neg p \vee \neg r \\
\mathbf{true} & \Rightarrow \neg u \vee \neg l \vee \neg q \\
\mathbf{true} & \Rightarrow \neg u \vee \neg l \vee w^{\mathbf{A}}_{\neg l} \\
w^{\mathbf{A}}_{\neg l} & \Rightarrow \mathbf{A}\bigcirc\big(\neg l \vee w^{\mathbf{A}}_{\neg l}\big).
\end{array}$$

Similar to ERES1, the intuition underlying the following ERES2 rule is to resolve an eventuality $\mathbf{E}_{\langle ind\rangle}\Diamond\neg l$, which states that $\Diamond\neg l$ is true on a path $\chi^{\langle ind\rangle}_{s_i}$, with a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, which, together, provided that their combined left-hand sides were true, imply that $\Box l$ also holds on the path $\chi^{\langle ind\rangle}_{s_{i+1}}$.

ERES2

$$\frac{\begin{array}{c}P^\dagger \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(\mathbf{E}_{\langle ind\rangle}\Box l) \\ Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l\end{array}}{Q \Rightarrow \mathbf{E}_{\langle ind\rangle}(\neg(P^\dagger)\,\mathcal{W}\,\neg l)},$$

where $P^\dagger \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(\mathbf{E}_{\langle ind\rangle}\Box l)$ represents a set, $\Lambda^{ind}_{\mathbf{E}\Box}$, of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses which is analogous to the set $\Lambda_{\mathbf{E}\Box}$, but each $*$ is either empty or an operator in $\{\mathbf{A}\bigcirc, \mathbf{E}_{\langle ind\rangle}\bigcirc\}$ and for every i, $1 \le i \le n$, conditions (1) and (2) are provable. Furthermore, $P^\dagger = \bigvee_{i=1}^{n}\bigwedge_{j=1}^{m_i} P^i_j$. Again, conditions (1) and (2) ensure that the set $\Lambda^{ind}_{\mathbf{E}\Box}$ of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses implies the formula $P^\dagger \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(\mathbf{E}_{\langle ind\rangle}\Box l)$.

Similarly, we use an atomic proposition $w^{ind}_{\neg l}$ uniquely associated with $\mathbf{E}_{\langle ind\rangle}\Diamond\neg l$ to represent the resolvent of ERES2 as the following set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses.

$$\begin{array}{l}
\big\{w^{ind}_{\neg l} \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\big(\neg l \vee \bigvee_{j=1}^{m_i}\neg P^i_j\big) \mid 1 \le i \le n\big\} \\
\cup\ \big\{\mathbf{true} \Rightarrow \neg Q \vee \neg l \vee \bigvee_{j=1}^{m_i}\neg P^i_j \mid 1 \le i \le n\big\} \\
\cup\ \big\{\mathbf{true} \Rightarrow \neg Q \vee \neg l \vee w^{ind}_{\neg l},\ w^{ind}_{\neg l} \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\big(\neg l \vee w^{ind}_{\neg l}\big)\big\}
\end{array}$$

As for ERES1, the use of atomic propositions uniquely associated with $\mathbf{E}$-eventualities allows us to represent all resolvents by ERES2 using a fixed set of atomic propositions depending only on the initial set of clauses.

This completes the presentation of the resolution rules of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$. We now introduce some useful definitions which are needed when we discuss the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ further.

Definition 4.6 (*Saturation with Respect to Step Resolution Rules*). A set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses is *saturated* with respect to step resolution rules if all clauses that can be derived by an application of one of the step resolution rules SRES1 to SRES8 to premises in T are contained in T.

Definition 4.7 (*Saturation with Respect to* $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$). A set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses is *saturated* with respect to $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ if all clauses that can be derived by an application of a rule of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ to premises in T are contained in T.

Definition 4.8 (*Derivation*). A *derivation* from a set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses by $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ is a sequence $T_0, T_1, T_2, \ldots$ of sets of clauses such that $T = T_0$ and $T_{t+1} = T_t \cup R_t$, where R_t is a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses obtained as the conclusion of an application of a rule of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ to premises in T_t.

Definition 4.9 (*Refutation*). A *refutation* of a set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses (by $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$) is a derivation from T such that for some $i \geq 0$, T_i contains a *contradiction*, where a contradiction is either the formula **true** $\Rightarrow$ **false** or **start** $\Rightarrow$ **false**.

Definition 4.10 (*Termination*). A derivation *terminates* iff either a contradiction is derived or no new clauses can be derived by any further application of resolution rules.

Next, we present an example of a refutation of a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, which involves both step resolution and eventuality resolution rules.

Example 4.11. In Example 3.3, we have seen that application of our transformation rules to the CTL formula $\varphi = \mathbf{E}\Box\neg l \wedge \mathbf{A}\Diamond l$ results in the following satisfiability equivalent set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses.

$$\begin{array}{rrl}
1. & \mathbf{start} & \Rightarrow p_1 \\
2. & p_1 & \Rightarrow \mathbf{A}\Diamond l \\
3. & p_2 & \Rightarrow \mathbf{E}_{\langle 1 \rangle}\bigcirc p_2 \\
4. & \mathbf{true} & \Rightarrow \neg p_1 \vee p_2 \\
5. & \mathbf{true} & \Rightarrow \neg p_2 \vee \neg l
\end{array}$$

Using step resolution, eventuality resolution, and rewrite rules with the ordering $l \succ p_1 \succ p_2 \succ w^A_l$ and the selection function S mapping every propositional disjunction C to an empty set, we are able to generate the following derivation. Note in steps 7–9, we resolve the **A**-sometime clause (clause 2) with the set $\Lambda_{\mathbf{E}\Box}$ of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses containing clauses 3 and 5.

$$\begin{array}{rrll}
6. & \mathbf{start} & \Rightarrow p_2 & [1, 4, \mathrm{SRES5}] \\
7. & w^{\mathbf{A}}_l & \Rightarrow \mathbf{A}\bigcirc(l \vee \neg p_2) & [2, 3, 5, \mathrm{ERES1}] \\
8. & \mathbf{true} & \Rightarrow \neg p_1 \vee l \vee \neg p_2 & [2, 3, 5, \mathrm{ERES1}] \\
9. & \mathbf{true} & \Rightarrow \neg p_1 \vee l \vee w^{\mathbf{A}}_l & [2, 3, 5, \mathrm{ERES1}] \\
10. & w^{\mathbf{A}}_l & \Rightarrow \mathbf{A}\bigcirc\big(l \vee w^{\mathbf{A}}_l\big) & [2, 3, 5, \mathrm{ERES1}] \\
11. & \mathbf{true} & \Rightarrow \neg p_1 \vee \neg p_2 & [5, 8, \mathrm{SRES8}] \\
12. & \mathbf{start} & \Rightarrow \neg p_2 & [1, 11, \mathrm{SRES5}] \\
13. & \mathbf{start} & \Rightarrow \mathbf{false} & [6, 12, \mathrm{SRES4}]
\end{array}$$

In the preceding, the notation $[c_1, \ldots, c_n, \mathrm{ERESi}]$ indicates that we apply the eventuality resolution rule ERESi to the clauses $c_1, \ldots, c_n$. Therefore, we have proved that $\mathbf{E}\Box\neg l \wedge \mathbf{A}\Diamond l$ is unsatisfiable.

4.3. Loop Search

The expensive part of applying ERES1 and ERES2 is finding sets of step and global clauses which can serve as premises for these rules, that is, for a given literal l stemming from some eventuality, to find sets of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses $\Lambda_{\mathbf{E}\Box}$, or $\Lambda^{ind}_{\mathbf{E}\Box}$, satisfying conditions (1) and (2). Such sets of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses are also called ***E**-loops in l* and the formula $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} P^i_j$ is called a *loop formula*. Algorithms to find loops were first presented in Bolotov and Dixon [2000]. Two loop search algorithms are defined, called **A**-loop search algorithm and **E**-loop search algorithm. An **A**-loop search algorithm is not required for our calculus as an **E**-loop search algorithm is sufficient to find the premises for both ERES1 and ERES2. Therefore, we only present an **E**-loop search algorithm here, which is slightly different from the **E**-loop search algorithm in Bolotov and Dixon [2000] due to the presence of global clauses, an ordering, and a selection function we introduce into $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$. In Section 7, we will discuss in more detail why an **A**-loop search algorithm is not required in our setting.

The **E**-loop search algorithm makes use of the notion of *merged clauses* which are inductively defined as follows.

—Any global clause, **A**-step clause, and **E**-step clause is a merged clause.
—For some $\mathbf{P}$, $\mathbf{P} \in \{\mathbf{A}\bigcirc, \mathbf{E}_{\langle ind\rangle}\bigcirc\}$ or empty if $A_1 \Rightarrow \mathbf{P}B_1$ and $A_2 \Rightarrow \mathbf{P}B_2$ are a pair of merged clauses, then $(A_1 \wedge A_2) \Rightarrow \mathbf{P}(B_1 \wedge B_2)$ is also a merged clause.
—If $A_1 \Rightarrow B_1$, $A_2 \Rightarrow \mathbf{A}\bigcirc B_2$, and $A_3 \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc B_3$, are merged clauses, then so are $(A_1 \wedge A_2) \Rightarrow \mathbf{A}\bigcirc(B_1 \wedge B_2)$, $(A_1 \wedge A_3) \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(B_1 \wedge B_3)$, and $(A_2 \wedge A_3) \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(B_2 \wedge B_3)$.

***E**-loop Search Algorithm.* The algorithm takes as input a literal l, stemming either from an **A**-sometime clause $Q \Rightarrow \mathbf{A}\Diamond\neg l$ or from an **E**-sometime clause $Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$, and a set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, among which we search for premises for the eventuality resolution rules. We assume the set T is saturated with respect to step resolution rules, that is, the rules SRES1 to SRES8.

The algorithm proceeds by constructing a sequence $H_0, H_1, H_2, \ldots$ of formulae which approximate a loop formula. In more detail, the algorithm works as follows.

(1) Search in T for merged clauses of the form $X_j \Rightarrow Y_j$, $X_j \Rightarrow \mathbf{A}\bigcirc Y_j$, and $X_j \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc Y_j$ such that $Y_j \Rightarrow l$ is provable (in propositional logic). Assuming there are n_0 such clauses, we construct the first formula, H_0, as follows.
$$H_0 = \textstyle\bigvee_{j=1}^{n_0} X_j.$$
Simplify H_0 using boolean simplification. If $H_0 \equiv \mathbf{true}$ a loop is found, we return **true** and the algorithm terminates. If $H_0 \equiv \mathbf{false}$ (which can only be the case if $n_0 = 0$), then no loop formula can be found and we return **false**.
(2) Given a formula H_i, where $i \geq 0$, construct the next formula H_{i+1} by looking in T for merged clauses of the form $A_j \Rightarrow (B_j \wedge l)$, $A_j \Rightarrow \mathbf{A}\bigcirc(B_j \wedge l)$, or $A_j \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(B_j \wedge l)$ such that $B_j \Rightarrow H_i$ is provable (in propositional logic). Assuming there are n_{i+1} such merged clauses, we construct the formula H_{i+1} as follows.
$$H_{i+1} = \textstyle\bigvee_{j=1}^{n_{i+1}} A_j.$$
Simplify H_{i+1} using boolean simplification.
(3) Repeat the previous step until one of the following conditions is provable (in propositional logic).
   (a) $H_{i+1} \equiv \mathbf{true}$. A loop formula has been found. We return **true** and the algorithm terminates.
   (b) $H_{i+1} \equiv \mathbf{false}$ (i.e., $n_{i+1} = 0$). No loop formula can be found. We return **false** and the algorithm terminates.
   (c) $H_i \equiv H_{i+1}$. A loop formula has been found. We return H_{i+1} and the algorithm terminates.

If we try to apply an eventuality resolution rule to an **E**-sometime clause $Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$, then the input set T to the **E**-loop search algorithm consists of the set of all global and **A**-step clauses we currently have at our disposal plus all **E**-step clauses with index *ind*. If we try to apply an eventuality resolution rule to an **A**-sometime clause $Q \Rightarrow \mathbf{A}\Diamond\neg l$, then the input set T to the **E**-loop search algorithm consists of the set of all global, **A**-step clauses, and **E**-step clauses.

If the algorithm returns a formula $H_{i+1} \not\equiv \mathbf{false}$, then

—$H_{i+1} = \bigvee_{j=1}^{n_{i+1}} \bigwedge_{k=1}^{t_j} q_j^k$, for some literals q_j^k, $1 \leq j \leq n_{i+1}$, $1 \leq k \leq t_j$, and

```
1  procedure $main$(N)
2  // N is a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses
3  begin
4      New := $\{$C $|$ C is a determinate clause in N$\}$;
5      ST  := $\{$C $|$ C is a sometime clause in N$\}$;
6      Old := $\emptyset$;
7      do
8          Old := $resolution\_sres$(Old, New);
9          New := $\emptyset$;
10         if ($\bot \notin$ Old) then
11            foreach $\mathbf{A}$-sometime clause and $\mathbf{E}$-sometime clause C in ST
12                G := $resolution\_eres$(Old, C);
13                if (G $\neq$ $\emptyset$) then
14                   New := New $\cup$ G;
15                end if
16            end for
17            New := New$\backslash$Old;
18         end if
19     while ($\bot \notin$ Old and New $\neq \emptyset$)
20     $output$();
21 end
```

Fig. 1. A decision procedure.

—there exists the following set of clauses in T,

$$\begin{array}{ccc} P^1_1 \Rightarrow *C^1_1 & & P^l_1 \Rightarrow *C^l_1 \\ \vdots & & \vdots \\ P^1_{m_1} \Rightarrow *C^1_{m_1} & \cdots & P^l_{m_l} \Rightarrow *C^l_{m_l} \end{array}$$

such that these clauses satisfy conditions (1) and (2) of ERES1/ERES2 as well as the restrictions imposed on the form of $*$, and moreover,

$$\bigvee_{r=1}^{l} \bigwedge_{s=1}^{m_r} P^r_s \equiv \bigvee_{j=1}^{n_{i+1}} \bigwedge_{k=1}^{t_j} q^k_j.$$

The proof of the correctness of this algorithm can be found in Bolotov and Dixon [2000].

An important step in the algorithm is the task of "looking for merged clauses". We can use step resolution with ordering and selection to achieve this, see, for example, Zhang et al. [2009b].

4.4. A Decision Procedure

We present a decision procedure based on the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ for determining the satisfiability of a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses in Figure 1.

The procedure takes a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses `N` as input and then splits `N` into the set `New` of determinate clauses and the set `ST` of sometime clauses (lines 4 and 5, respectively). The set `Old` is initially set to be empty (line 6). We then enter the main loop of the procedure which will be repeated until either the contradiction $\bot$ (i.e., **start** $\Rightarrow$ **false** or **true** $\Rightarrow$ **false**) has been derived or we cannot derive any new clauses (line 7 to 19). We saturate the set `New`$\cup$`Old` using the step resolution rules and the resulting set of clauses becomes the set `Old` (line 8). If we have not derived the contradiction yet, then we try to apply eventuality resolution rules to each of the sometime clauses (lines 11 to 16). The union of all the resolvents generated by applications of the eventuality resolution rules becomes the set of new clauses `New`. Some of these resolvents may be redundant. Therefore, we eliminate clauses from `New` which are already in `Old` (line 17). Finally, after the main loop terminates, we print out the satisfiability of `N` (line 20).

4.5. Implementation

The calculus has been implemented as the prover CTL-RP. The implementation uses a classical, first-order resolution theorem prover, SPASS [Weidenbach et al. 2007] to implement the rules. In particular, all $\text{SNF}^{\text{g}}_{\text{CTL}}$ clauses, except the **A**-sometime clauses and **E**-sometime clauses, are transformed into first-order clauses. Then first-order ordered resolution with selection [Bachmair and Ganzinger 2001] is used to emulate step resolution. However, **A**-sometime clauses and **E**-sometime clauses cannot be translated into first-order logic. To apply the rules ERES1 and ERES2 for inferences with **A**-sometime clauses and **E**-sometime clauses, respectively, we use the loop search algorithm presented in Section 4.3 to find suitable premises. We can again utilise first-order ordered resolution with selection to perform the task of looking for merged clauses in the loop search algorithm, and we compute the results of applications of the eventuality resolution rules in the form of first-order clauses. In general, we would expect CTL-RP to perform better on unsatisfiable than satisfiable problems because, for the latter, the clause set must be saturated before it can terminate. Similarly we would expect it to have more difficulty on problems with a large number of complex eventualities.

More details about CTL-RP are given in Zhang et al. [2009b, Zhang 2010], along with an experimental comparison with a tableau prover for CTL (namely, the CTL module for the Tableau Workbench) [Abate and Goré 2003]. We will refer to the latter as TWB-CTL. As far as we know, there is no implementation of the calculus in Bolotov [2000]. There are no standard benchmarks for CTL provers so four types of formulae are considered: standard CTL equivalences; formulae representing a small, hand-crafted, finite state transition system; randomly generated state transition systems; and a specification of the Alternating Bit Protocol (see, e.g., [Huth and Ryan 2004]). Both provers solve the standard CTL equivalencies in 0.01 seconds or less. CTL-RP outperforms TWB-CTL for the handcrafted and random state transition systems in terms of the time taken to solve problems and the number solved within the specified time limit. CTL-RP completes all the Alternating Bit protocol problems, whereas TWB-CTL does not terminate on any of these problems.

In Goré et al. [2011], a comparison of five CTL provers, including CTL-RP, is carried out. The provers compared are CTL-RP, BDDCTL [Marrero 2005], MLSolver [Friedmann and Lange 2009], GMUL [Ben-Ari et al. 1981], and TreeTab [Abate et al. 2007]. BDDCTL combines a Hintikka-style approach with ordered binary decision digrams. MLSover is a satiability solver for modal fixed point calculi. GMUL and TreeTab are two-pass and one-pass tableau algorithms, respectively. Each of these is discussed more fully in Section 7. Goré et al. [2011] conclude that no one prover is the best for all problems and that each prover has problems where it performed badly. Additionally, they state that CTL-RP (and BDDCTL) are "more robust than the tableaux methods since they tend to succeed eventually rather than fail spectacularly or succeed spectacularly". They also note implementation flaws in BDDCTL. CTL-RP is the best or second best performer for the problems in the category Alternating Bit Protocol and for three of the formulae sets in the category of "step formulae" from Marrero [2005]. Additionally, it is the best performer in the satisfiable Montali's formulae category.

5. CORRECTNESS OF THE CALCULUS $\mathbf{R}^{\succ,S}_{\mathbf{CTL}}$

Next, we consider the correctness of the transformation into normal form, and the soundness, completeness, and termination of the calculus.

5.1. Correctness of the Transformation to $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$

In the following, we show that our transformation

(1) preserves satisfiability,
(2) is terminating, and
(3) allows only a polynomially bounded number of transformation rule application.

The proofs of that each transformation rule preserves satisfiability have been moved to the Electronic Appendix.

THEOREM 5.1. *Let $T_t = \Delta \cup \{\psi\}$ and $T_{t+1} = \Delta \cup R_t$ be two sets of CTL clauses such that T_{t+1} is obtained by an application of a transformation rule of the form $\psi \rightarrow R_t$ in the set Trans to the formula ψ in T_t. Then T_t is satisfiable iff T_{t+1} is satisfiable.*

PROOF. To prove this theorem, we need to show that every transformation rule in the set *Trans* preserves satisfiability.

Lemmas A.4, A.5, A.6, A.7, A.8, and A.9 prove that the transformation rules *Trans*(1), *Trans*(3), *Trans*(5), *Trans*(6), *Trans*(10), and *Trans*(11) preserve satisfiability, respectively.

The other operators and rules not proved explicitly are similar to the cases shown. The proofs for *Trans*(2) are similar to *Trans*(1), for *Trans*(4), *Trans*(7), *Trans*(8) and *Trans*(9) are similar to *Trans*(6) and for *Trans*(12) are similar to *Trans*(11). □

The definitions relating to and the proof that each application of a transformation rule to a clause Γ in a set T of CTL clauses results in a set T' of CTL clauses weighs strictly less than T can be found in the Electronic Appendix.

THEOREM 5.2. *Let $T_0, T_1, \ldots$ be a sequence of sets of CTL clauses such that $T_0 = init(\varphi)$ for some CTL formula φ and T_{t+1} is obtained from T_t by applying a transformation rule to a clause in T_t. Then the sequence $T_0, T_1, \ldots$ terminates, that is, there exists an index $n, n \geq 0$, such that no transformation rule can be applied to any clause in T_n. Furthermore, all clauses in T_n are in $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$.*

PROOF. Follows from Lemma A.17 and Theorem A.16. □

Using the following notation of the size of a CTL formula, we are able to characterise the computational complexity of the normal form transformation.

Definition 5.3 (*Size of a CTL Formula*). Let φ and ψ be arbitrary CTL formulae and p be an arbitrary atomic proposition in P_{PL}. We inductively define the size *sz* of an arbitrary CTL formula as follows:

(1) $sz(\mathbf{true}) = sz(\mathbf{false}) = sz(p) = 1$;
(2) $sz(\neg\varphi) = sz(\mathbf{A}\square\varphi) = sz(\mathbf{A}\diamond\varphi) = sz(\mathbf{A}\bigcirc\varphi) = sz(\mathbf{E}\square\varphi) = sz(\mathbf{E}\diamond\varphi) = sz(\mathbf{E}\bigcirc\varphi) = sz(\varphi) + 1$; and
(3) $sz(\varphi \wedge \psi) = sz(\varphi \vee \psi) = sz(\varphi \Rightarrow \psi) = sz(\mathbf{A}(\varphi\,\mathcal{U}\,\psi)) = sz(\mathbf{A}(\varphi\,\mathcal{W}\,\psi)) = sz(\mathbf{E}(\varphi\,\mathcal{U}\,\psi)) = sz(\mathbf{E}(\varphi\,\mathcal{W}\,\psi)) = sz(\varphi) + sz(\psi) + 1$.

THEOREM 5.4. *Let φ be an arbitrary CTL formula and T_n be a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses obtained from $T_0 = init(\varphi)$ by n applications of our transformation rules. Then n is linearly bounded in the size of φ and the set T_n can be computed in polynomial time in the size of φ.*

PROOF. By Lemma A.18, we show that T_n can be computed in less than $47m + 9$ applications of the transformation rules, where m is the size of φ.

Regarding the complexity of each application, we assume CTL clauses are stored in a tree data structure. Then according to our transformation rules, by reusing the subtrees representing subformulae as appropriate, generating the results of clauses from the clause which the rule applies to can be accomplished in constant time in the size of φ. The pattern matching procedure determining that for a given CTL clause not in $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ which rule to apply, requires linear time in the size of φ in the worst case.

Therefore, the set T_n can be computed in polynomial time in the size of φ. □

THEOREM 5.5. *Let φ be an arbitrary CTL formula and T_n be a set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses obtained from $T_0 = init(\varphi)$ by a linearly bounded applications of our transformation rules in the size of φ. Then φ is satisfiable iff T_n is satisfiable.*

PROOF. Follows from Theorem 5.2, Lemma A.2, Theorem 5.1, and Theorem 5.4. □

5.2. Soundness and Completeness

THEOREM 5.6 (SOUNDNESS OF $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$). *Let T be a set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses. If there is a refutation of T by $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$, then T is unsatisfiable.*

PROOF. Let $T_0, T_1, \ldots, T_n$ be a derivation from a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clause $T_0 = T$ by the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$. We will show by induction over the length of the derivation that if T_0 is satisfiable, then so is T_n.

For $T_0 = T$, the claim obviously holds. Now, consider the step of the derivation in which we derive T_{t+1} from T_t for some $t \geq 0$. Assume T_t is satisfiable and $M = \langle S, R, L, [\_], s_0\rangle$ is a model structure satisfying T_t.

We show that SRES2 is sound. Assume $\mathbf{A}\square(P \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee l))$ and $\mathbf{A}\square(Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l))$ are in T_t. Let T_{t+1} be obtained by an application of SRES2 to $\mathbf{A}\square(P \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee l))$ and $\mathbf{A}\square(Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l))$, that is, $T_{t+1} = T_t \cup \{\mathbf{A}\square(P \wedge Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee D))\}$. We show that M also satisfies T_{t+1}. Consider an arbitrary state $s \in S$. If $M, s \not\models P$ or $M, s \not\models Q$, then obviously $M, s \models P \wedge Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee D)$. Assume that $M, s \models P$ and $M, s \models Q$. From $\mathbf{A}\square(P \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee l))$, $\mathbf{A}\square(Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l))$ and the semantics of $\mathbf{A}\square$, we obtain that $M, s \models P \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee l)$ and $M, s \models Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l)$. From the semantics of $\Rightarrow$, we obtain that $M, s \models \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee l)$ and $M, s \models \mathbf{A}\bigcirc(D \vee \neg l)$. From the semantics of $\mathbf{A}\bigcirc$, we obtain that for all successors s' of state s, $M, s' \models D \vee \neg l$. From the semantics of $\mathbf{E}_{\langle ind\rangle}\bigcirc$, we have that for the successor s'' of s such that s'' is on the path $\chi_s^{\langle ind\rangle}$, $M, s'' \models C \vee l$. As s'' is a successor of s, $M, s'' \models D \vee \neg l$. As l and $\neg l$ cannot both be true at state s'', we conclude that $M, s'' \models C \vee D$. From the semantics of $\mathbf{E}_{\langle ind\rangle}\bigcirc$, we have $M, s \models \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee D)$. Therefore, $M, s \models P \wedge Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee D)$. As s is arbitrary, from the semantics of $\mathbf{A}\square$, we have $M, s_0 \models \mathbf{A}\square(P \wedge Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(C \vee D))$.

The cases for SRES1, SRES5, and SRES6 are shown in the Electronic Appendix. For the rule SRES3, the proof is analogous to that for SRES2; for the rules SRES4 and SRES8, the proofs are analogous to that for SRES5; and for the rule SRES7, the proof is analogous to that for SRES6.

Regarding RW1, from the semantics of $\mathbf{A}\bigcirc$ and **false**, we obtain that the formula $\mathbf{A}\square(Q \Rightarrow \mathbf{A}\bigcirc\mathbf{false})$ is true iff $\mathbf{A}\square(Q \Rightarrow \mathbf{false})$ is true. This formula is propositionally equivalent to $\mathbf{A}\square(\neg Q)$ which in turn, by the semantics of $\Rightarrow$ and **true**, is equivalent to $\mathbf{A}\square(\mathbf{true} \Rightarrow \neg Q)$. The proof for RW2 is analogous.

Next, we show ERES1 is sound. Assume that $\mathbf{A}\square(Q \Rightarrow \mathbf{A}\diamond\neg l)$ is in T_t and there exists a set $\Lambda_{\mathbf{E}\square}$ of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses in T_t together implying $\mathbf{A}\square(P^{\dagger} \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\square l)$, where $P^{\dagger}$ is a disjunction of conjunctions of literals (defined in Section 4.2). Therefore, M satisfies $\mathbf{A}\square(P^{\dagger} \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\square l)$. We show that M also satisfies $\mathbf{A}\square(Q \Rightarrow \mathbf{A}(\neg(P^{\dagger})\,\mathcal{W}\,\neg l))$. Consider an arbitrary state $s_i \in S$. If $M, s_i \not\models Q$, then obviously $M, s_i \models Q \Rightarrow \mathbf{A}(\neg(P^{\dagger})\,\mathcal{W}\,\neg l)$. Assume $M, s_i \models Q$. From $\mathbf{A}\square(Q \Rightarrow \mathbf{A}\diamond\neg l)$ and the semantics of $\mathbf{A}\square$ and $\Rightarrow$, we obtain that $M, s_i \models \mathbf{A}\diamond\neg l$. If $M, s_i \models \neg l$, then by the semantics of $\mathbf{A}\mathcal{W}$, $M, s_i \models \mathbf{A}(\neg(P^{\dagger})\,\mathcal{W}\,\neg l)$. From the semantics of $\Rightarrow$, $M, s_i \models Q \Rightarrow \mathbf{A}(\neg(P^{\dagger})\,\mathcal{W}\,\neg l)$. If, on the other hand, $M, s_i \models l$, then from the semantics of $\mathbf{A}\Diamond$, we know that for every path χ_{s_i}, there exists $s_j \in \chi_{s_i}, j > i$ such that $M, s_j \models \neg l$ and for every $k, i \leq k < j, M, s_k \models l$.

For an arbitrary state $s \in S$, if $M, s \models \mathbf{A}\Diamond\neg l \wedge l$, then by the semantics of $\mathbf{A}\bigcirc$ and $\mathbf{A}\Diamond$, $M, s \models \mathbf{A}\bigcirc\mathbf{A}\Diamond\neg l$. Therefore, for all the successors s' of s, $M, s' \models \mathbf{A}\Diamond\neg l$.

Due to this property and $M, s_i \models \mathbf{A}\Diamond\neg l \wedge l$ and $M, s_k \models l$, by an inductive augment we can conclude that for every $k, i \leq k < j, M, s_k \models \mathbf{A}\Diamond\neg l$. As we know, $M, s_k \models l$. Therefore, $M, s_k \models \mathbf{A}\Diamond\neg l \wedge l$. From the semantics of $\mathbf{A}\Diamond$, for all paths χ_{s_k}, there exists $s_n \in \chi_{s_k}, n > k, M, s_n \models \neg l$. Next, we use a proof by contradiction to establish that for all $k, i \leq k < j, M, s_k \models \neg(P^\dagger)$. Assume that $P^\dagger$ holds at s_k. From the semantics of $\mathbf{A}\Box$ and $\Rightarrow$, we have $M, s_k \models \mathbf{E}\bigcirc\mathbf{E}\Box l$. From the semantics of $\mathbf{E}\bigcirc$ and $\mathbf{E}\Box$, we know that there exists a path χ_{s_k} such that for all states $s_m \in \chi_{s_k}, m > k, M, s_m \models l$. This is a contradiction. Therefore, $\neg(P^\dagger)$ must hold at all the states s_k. From the semantics of $\mathbf{A}\mathcal{U}$, we obtain that $M, s_i \models \mathbf{A}(\neg(P^\dagger)\,\mathcal{U}\,\neg l)$. From the semantics of $\mathbf{A}\mathcal{W}$, we obtain that $M, s_i \models \mathbf{A}(\neg(P^\dagger)\,\mathcal{W}\,\neg l)$. Thus, $M, s_i \models Q \Rightarrow \mathbf{A}(\neg(P^\dagger)\,\mathcal{W}\,\neg l)$. As s_i is arbitrary, from the semantics of $\mathbf{A}\Box$, $M, s_0 \models \mathbf{A}\Box(Q \Rightarrow \mathbf{A}(\neg(P^\dagger)\,\mathcal{W}\,\neg l))$. The proof for ERES2 is analogous. $\Box$

Our proof of the completeness of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ makes use of (reduced) labelled behaviour graphs, which will be defined later in this section. These graphs can be seen as finite representations of the set of all models of a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses.

First, we briefly discuss how our proof proceeds. We introduce the idea of augmentation, which was originally developed for a resolution calculus for PLTL [Fisher et al. 2001]. Next, we create a finite labelled directed graph, called a labelled behaviour graph, for an augmented set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses. However, some nodes and some subgraphs of the labelled behaviour graph for T cannot be used to create a CTL model structure for a set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses. For instance, a node without any successor nodes in a labelled behaviour graph cannot be used to construct a CTL model structure, as all paths in a CTL model structure are infinite. To remove such nodes and subgraphs from a labelled behaviour graph, we define a set of deletion rules. We call a labelled behaviour graph H a reduced labelled behaviour graph if it is obtained by exhaustively applying deletion rules to H. We show that, if an augmented set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses is unsatisfiable, then its reduced labelled behaviour graph is empty. We also prove that each application of a deletion rule corresponds to a derivation from T by $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$. Therefore, if T is unsatisfiable, its reduced labelled behaviour graph H_{red} is empty, and the sequence of applications of the deletion rules, which reduce the labelled behaviour graph for T to an empty H_{red}, can be used to construct a refutation in $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$. In the following, we show the detailed completeness proof.

Let T be a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses obtained by applying the normal form transformation to a given CTL formula. Recall from Section 4.2 an application of ERES1 or ERES2 to the set T may introduce new propositions, for example, $w^{\mathbf{A}}_{\neg l}$ and $w^{ind}_{\neg l}$, into T. Our completeness proof makes use of the labelled behaviour graph, whose construction depends on the set $\mathit{Prop}(T)$ of propositions occurring in T. Due to this, we do not want $\mathit{Prop}(T)$ to change during a derivation. Therefore, we introduce the notion of *augmentation*, which adds clauses associated with these new propositions into T right from the beginning, that is, before any resolution rules are applied to T. In this way, we can be sure that no new propositions appear during the application of the resolution rules. That is, $\mathit{Prop}(T)$ stays the same if T is augmented. Moreover, we also show that augmentation is correct.

We adapt the *augmentation* procedure used in Fisher et al. [2001] for PLTL to CTL to establish a relation between the new atomic propositions introduced by applications of ERES1 or ERES2 and eventualities associated with them.

Definition 5.7 (*Augmentation*). Given a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clause T, we construct an augmented set $aug(T)$ as follows: the augmented set $aug(T)$ is the smallest set containing T and satisfying the following conditions.

—For every **A**-sometime clause in T, $Q \Rightarrow \mathbf{A}\Diamond\neg l$, $aug(T)$ contains the clauses

$$\begin{aligned}\mathbf{true} &\Rightarrow \neg Q \vee \neg l \vee w^{\mathbf{A}}_{\neg l}\\ w^{\mathbf{A}}_{\neg l} &\Rightarrow \mathbf{A}\bigcirc\big(\neg l \vee w^{\mathbf{A}}_{\neg l}\big),\end{aligned}$$

where $w^{\mathbf{A}}_{\neg l}$ is the proposition uniquely associated with $\mathbf{A}\Diamond\neg l$ (i.e., $w^{\mathbf{A}}_{\neg l}$ is the same proposition we used for ERES1).

—For every **E**-sometime clause in T, $Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$, $aug(T)$ contains the clauses

$$\begin{aligned}\mathbf{true} &\Rightarrow \neg Q \vee \neg l \vee w^{ind}_{\neg l}\\ w^{ind}_{\neg l} &\Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\big(\neg l \vee w^{ind}_{\neg l}\big),\end{aligned}$$

where $w^{ind}_{\neg l}$ is the proposition uniquely associated with $\mathbf{E}_{\langle ind\rangle}\Diamond l$ (i.e., $w^{ind}_{\neg l}$ is the same proposition we used for ERES2).

Next, we formally prove that our augmentation preserves satisfiability.

LEMMA 5.8. *Let T be a set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses and M be a model structure satisfying T. If T' is a subset of T, then M also satisfies T'.*

PROOF. Straightforward. □

LEMMA 5.9. *Let T be a set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses. The augmented set $aug(T)$ is satisfiable iff T is satisfiable.*

PROOF. As $T \subset aug(T)$, by Lemma 5.8 if $aug(T)$ is satisfiable and a model structure M satisfies $aug(T)$, then M also satisfies T, and thus, T is satisfiable.

Conversely, if T holds in a model structure M_1 at the state s_0, then M_1 can be extended to another model structure M_2 by giving $w^{\mathbf{A}}_{\neg l}$ the same truth value as $l \wedge \mathbf{A}\Diamond\neg l$ and $w^{ind}_{\neg l}$ the same truth value as $l \wedge \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$ in each state in M_2 for each eventuality $\neg l$ in T. We show that M_2 satisfies $aug(T)$ at the state s_0 of M_2.

Assume that $Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$ is in T. We show that M_2 satisfies $\mathbf{A}\Box(\mathbf{true} \Rightarrow \neg Q \vee \neg l \vee w^{ind}_{\neg l})$ and $\mathbf{A}\Box(w^{ind}_{\neg l} \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(\neg l \vee w^{ind}_{\neg l}))$, that is, the two clauses added by augmentation for $Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$. Let s be an arbitrary state in M_2.

(1) We know that $M_1, s_0 \models \mathbf{A}\Box(Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l)$. By the definition of M_2 and Lemma A.1, we know that $M_2, s_0 \models \mathbf{A}\Box(Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l)$. From the semantics of $\mathbf{A}\Box$, $M_2, s \models Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$. By propositional reasoning, $M_2, s \models (Q \wedge l) \Rightarrow (l \wedge \mathbf{E}_{\langle ind\rangle}\Diamond\neg l)$. By the definition of M_2, $M_2, s \models (l \wedge \mathbf{E}_{\langle ind\rangle}\Diamond\neg l) \Rightarrow w^{ind}_{\neg l}$. Therefore, by the semantics of $\Rightarrow$, we obtain that $M_2, s \models (Q \wedge l) \Rightarrow w^{ind}_{\neg l}$. Thus, $M_2, s \models \mathbf{true} \Rightarrow \neg Q \vee \neg l \vee w^{ind}_{\neg l}$. Since s is an arbitrary state in M_2, from the semantics of $\mathbf{A}\Box$, we obtain that $M_2, s_0 \models \mathbf{A}\Box(\mathbf{true} \Rightarrow \neg Q \vee \neg l \vee w^{ind}_{\neg l})$.

(2) From the definition of M_2, we know $M_2, s \models w^{ind}_{\neg l} \Rightarrow (l \wedge \mathbf{E}_{\langle ind\rangle}\Diamond\neg l)$.
—If $M_2, s \not\models w^{ind}_{\neg l}$, then $M_2, s \models w^{ind}_{\neg l} \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(\neg l \vee w^{ind}_{\neg l})$.
—If, on the other hand, $M_2, s \models w^{ind}_{\neg l}$, then $M_2, s \models l \wedge \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$. Thus, from the semantics of $\mathbf{E}_{\langle ind\rangle}\Diamond$, for the state s' with $(s, s') \in [ind]$, either $M_2, s' \models \neg l$ or $M_2, s' \models l \wedge \mathbf{E}_{\langle ind\rangle}\Diamond\neg l$. By the definition of M_2, we know that $M_2, s' \models (l \wedge \mathbf{E}_{\langle ind\rangle}\Diamond\neg l) \Rightarrow w^{ind}_{\neg l}$. Thus, either $M_2, s' \models \neg l$ or $M_2, s' \models w^{ind}_{\neg l}$. From the semantics of $\mathbf{E}_{\langle ind\rangle}\bigcirc$ and $\vee$, $M_2, s \models \mathbf{E}_{\langle ind\rangle}\bigcirc(\neg l \vee w^{ind}_{\neg l})$. As $M_2, s \models w^{ind}_{\neg l}$, we obtain that $M_2, s \models w^{ind}_{\neg l} \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(\neg l \vee w^{ind}_{\neg l})$.

As s is arbitrary, from the semantics of $\mathbf{A}\Box$, $M_2, s_0 \models \mathbf{A}\Box(w^{ind}_{\neg l} \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(\neg l \vee w^{ind}_{\neg l}))$.

The proof for $Q \Rightarrow \mathbf{A}\diamond\neg l$ in T is analogous. Therefore, $aug(T)$ is satisfied in M_2 at state s_0. □

We now introduce the notion of a *labelled behaviour graph*. Given a set *Ind* of indices, an *Ind-labelled graph H* is an ordered pair $H = (N, E)$, where N is a set of nodes and E is a set of directed edges in H of the form (n, ind, n'), where $n, n' \in N$ and $ind \in Ind$. If there exists an edge $(n, ind, n') \in E$ for some $ind \in Ind$, then n' is a *successor* of n and n is a *predecessor* of n'. If the label ind is also important for the relation of n and n' in the context, we also say that n' is an *ind-successor* of n and n is an *ind-predecessor* of n'. When the label on the edge is not important, we use (n, n') to denote an edge, which means the label can be any index in *Ind*.

Definition 5.10 (*Ind-Reachable Node in a Graph*). Given a set *Ind* of indices, an *Ind*-Labelled Graph (N, E), and a node $n \in N$, a node $n' \in N$ is *ind*-reachable from n iff there exists an edge $(n, ind, n') \in E$ or there exists an edge $(n'', ind, n') \in E$ and n'' is *ind*-reachable from n.

Definition 5.11 (*Reachable Node in a Graph*). Given a graph (N, E) and a node $n \in N$, a node $n' \in N$ is reachable from n iff there exists an edge $(n, n') \in E$ or there exists an edge $(n'', n') \in E$ and n'' is reachable from n.

Definition 5.12 (*Labelled Behaviour Graph*). Let T be an augmented set of $\text{SNF}^{\text{g}}_{\text{CTL}}$ clauses and $\text{Ind}(T)$ be the set of indices occurring in T. If $\text{Ind}(T)$ is empty, then let $\text{Ind}(T) = \{ind\}$, where ind is an arbitrary index in Ind. Given T and $\text{Ind}(T)$, we construct a finite directed graph $G = (N, E)$ for T as follows.

The set of nodes N of G consists of all ordered tuples $n = (V, E_A, E_E)$, where

(1) V is a valuation of the atomic propositions occurring in T;
(2) E_A is a subset of $\{l \mid Q \Rightarrow \mathbf{A}\diamond l \in T\}$; and
(3) E_E is a subset of $\{l_{\langle ind\rangle} \mid Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\diamond l \in T\}$.

Informally E_A and E_E contain eventualities that need to be satisfied either in the current node or some node reachable from the current node.

To define the set of edges E of G, we use the following auxiliary definitions. Let $n = (V, E_A, E_E)$ be a node in N. Let $R_A(n, T) = \{D \mid Q \Rightarrow \mathbf{A}\bigcirc D \in T$, and $V \models Q\}$. Note if V does not satisfy the left-hand side of any $\mathbf{A}$-step clause (i.e., $R_A(n, T) = \emptyset$), then there are no constraints from $\mathbf{A}$-step clauses on successor node of the node n and any valuation satisfies $R_A(n, T)$. Let $R_{ind}(n, T) = \{D \mid Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc D \in T$ and $V \models Q\}$. Let $R_g(T) = \{D \mid \mathbf{true} \Rightarrow D \in T\}$.

Let functions $\mathsf{Ev}_{\mathbf{A}}(V, T)$ and $\mathsf{Ev}_{\mathbf{E}}(V, T)$ be defined as

$$\mathsf{Ev}_{\mathbf{A}}(V, T) = \{l \mid Q \Rightarrow \mathbf{A}\diamond l \in T \text{ and } V \models Q\},$$

and

$$\mathsf{Ev}_{\mathbf{E}}(V, T) = \left\{l_{\langle ind\rangle} \mid Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\diamond l \in T \text{ and } V \models Q\right\},$$

respectively. Let functions $\mathsf{Unsat}_{\mathbf{A}}(E_A, V)$ and $\mathsf{Unsat}_{ind}(E_E, V)$ be defined as

$$\mathsf{Unsat}_{\mathbf{A}}(E_A, V) = \left\{l \mid l \in E_A \text{ and } V \not\models l\right\},$$

and

$$\mathsf{Unsat}_{ind}(E_E, V) = \left\{l_{\langle ind\rangle} \mid l_{\langle ind\rangle} \in E_E \text{ and } V \not\models l\right\},$$

respectively. For a node $n = (V, E_A, E_E)$ in G, if $l \in E_A (l_{ind} \in E_E)$ and $V \models l$, then we say that l (l_{ind}) is satisfied in node n.

Then E contains an edge labelled by ind from a node (V, E_A, E_E) to a node (V', E'_A, E'_E) iff V' satisfies the set $R_A(n, T) \cup R_{ind}(n, T) \cup R_g(T)$, $E'_A = \mathsf{Unsat}_{\mathbf{A}}(E_A, V) \cup \mathsf{Ev}_{\mathbf{A}}(V', T)$

and $E'_E = \mathsf{Unsat}_{ind}(E_E, V) \cup \mathsf{Ev}_{\mathbf{E}}(V', T)$. That is, there is an edge labelled with *ind* from (V, E_A, E_E) to (V', E'_A, E'_E) iff (i) V' satisfies all constraints imposed by **A**-step clauses, **E**-step clauses with the index *ind*, and global clauses whose left-hand sides are satisfied by V, (ii) E'_A consists of **A**-eventualities not satisfied by V plus additional **A**-eventualities triggered by V', and (iii) E'_E consists of **E**-eventualities with the index *ind* not satisfied by V plus additional **E**-eventualities with the index *ind* triggered by V'.

Let $R_0(T) = \{D \mid \mathbf{start} \Rightarrow D \in T\}$. Then the node (V, E_A, E_E), where V satisfies the set $R_0(T) \cup R_g(T)$, $E_A = \mathsf{Ev}_{\mathbf{A}}(V, T)$ and $E_E = \mathsf{Ev}_{\mathbf{E}}(V, T)$, is an initial node of G. That is, initial nodes are those nodes such that (i) V satisfies all constraints imposed by initial clauses and global, (ii) E_A consists of **A**-eventualities triggered by V, and (iii) E_E consists of **E**-eventualities with the index *ind* triggered by V.

The *labelled behaviour graph* $H = (N', E')$ for an augmented set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses T is the subgraph of G such that the set $N' \subseteq N$ of nodes and the set $E' \subseteq E$ of edges are reachable from the initial nodes of G.

Definition 5.13 (*Path from a Node n to a Node n′ through a Graph*). A path from a node n_1 to a node n_k in a graph is a sequence of nodes $n_1, n_2, \ldots, n_k$ such that $(n_1, n_2), (n_2, n_3), \ldots, (n_{k-1}, n_k)$ are edges of the graph.

Definition 5.14 (*Shortest Path from a Node n to a Node n′ through a Graph*). A shortest path from a node n to a node n' in a graph is a path from the node n to the node n' with the least number of edges amongst all the paths from the node n to the node n'.

Definition 5.15 (*Distance*). Given a graph (N, E), if a node $n' \in N$ is reachable from another node $n \in N$, the distance from n to n' is the number of edges in a shortest path from n to n'.

Definition 5.16 (*Ind-Distance*). Given a graph (N, E), if a node $n' \in N$ is *ind*-reachable from a node $n \in N$, the *ind*-distance from n to n' is the number of edges in a shortest path such that every edge in it is labelled by *ind*.

Lemma 5.17. *Let T be an augmented set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses and $H = (N, E)$ be the labelled behaviour graph for T. If H contains an edge from a node $n = (V, E_A, E_E) \in N$ to a node $n' = (V', E'_A, E'_E) \in N$ such that $l \in E'_A$ then either (i) there exists a clause $Q \Rightarrow \mathbf{A}\Diamond l \in T$ such that $V' \models Q$ or (ii) $l \in E_A$ and $V \not\models l$.*

Proof. From the construction of the labelled behaviour graph, we know $E'_A = \mathsf{Unsat}_{\mathbf{A}}(E_A, V) \cup \mathsf{Ev}_{\mathbf{A}}(V', T)$. Therefore, if $l \in E'_A$, then l is either from $\mathsf{Unsat}_{\mathbf{A}}(E_A, V)$ or from $\mathsf{Ev}_{\mathbf{A}}(V', T)$. For the first case, l must be in E_A and $V \not\models l$. For the latter, there exists a clause $Q \Rightarrow \mathbf{A}\Diamond l \in T$ such that $V' \models Q$. □

Lemma 5.18. *Let T be an augmented set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses and $H = (N, E)$ be the labelled behaviour graph for T. Then, for every node $n = (V, E_A, E_E)$ in H, if $l \in E_A$ and $V \not\models l$ then $V \models w^{\mathbf{A}}_l$.*

Proof. The proof proceeds by induction over the nodes of a path $(n_0, n_1, \ldots)$ from an initial node $n_0 = (V^0, E^0_A, E^0_E)$ to the node $n = (V, E_A, E_E)$.

In the base case, n is an initial node. If $l \in E^0_A$, by the construction of initial nodes, there must be an **A**-sometime clause $Q \Rightarrow \mathbf{A}\Diamond l \in T$ and $V^0 \models Q$. By augmentation, $\mathbf{true} \Rightarrow \neg Q \vee l \vee w^{\mathbf{A}}_l$. If $V^0 \not\models l$, we obtain that $V^0 \models w^{\mathbf{A}}_l$.

Otherwise, we assume that the lemma holds from node n_0 to $n_i = (V^i, E^i_A, E^i_E)$, $i > 0$, and we prove that it holds for node $n_{i+1} = (V^{i+1}, E^{i+1}_A, E^{i+1}_E)$. Based on the assumption of the lemma, $V^{i+1} \not\models l$ and $l \in E^{i+1}_A$. By Lemma 5.17, since $l \in E^{i+1}_A$, either

(1) there exists a clause $Q \Rightarrow \mathbf{A}\Diamond l \in T$ such that $V^{i+1} \models Q$ or
(2) $l \in E^i_A$ and $V^i \not\models l$.

In case (1), by augmentation, **true** $\Rightarrow \neg Q \vee l \vee w^{\mathbf{A}}_l \in T$, and since $V^{i+1} \not\models l$, we obtain that $V^{i+1} \models w^{\mathbf{A}}_l$. In case (2) by the induction hypothesis we have $V^i \models w^{\mathbf{A}}_l$. By augmentation we have $w^{\mathbf{A}}_l \Rightarrow \mathbf{A}\bigcirc(w^{\mathbf{A}}_l \vee l) \in T$. Thus by the construction of H, since $V^{i+1} \not\models l$, we have $V^{i+1} \models w^{\mathbf{A}}_l$. Thus, the lemma also holds for node n_{i+1}. □

LEMMA 5.19. *Let T be an augmented set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses and $H = (N, E)$ be the labelled behaviour graph for T. If H contains an edge from a node $n = (V, E_A, E_E) \in N$ to a node $n' = (V', E'_A, E'_E) \in N$ such that $l_{\langle ind\rangle} \in E'_E$, then either (i) there exists a clause $Q \Rightarrow \boldsymbol{E}\Diamond l_{\langle ind\rangle} \in T$ such that $V' \models Q$ or (ii) $l_{\langle ind\rangle} \in E_E$ and $V \not\models l$.*

PROOF. By the construction of the labelled behaviour graph, $E'_E = \mathsf{Unsat}_{ind}(E_E, V) \cup \mathsf{Ev}_{\mathbf{E}}(V', T)$. Therefore, if $l_{\langle ind\rangle} \in E'_E$, then $l_{\langle ind\rangle}$ is either from $\mathsf{Unsat}_{ind}(E_E, V)$ or from $\mathsf{Ev}_{\mathbf{E}}(V', T)$. For the first case, $l_{\langle ind\rangle}$ must be in E_E and $V \not\models l$. For the latter, there exists a clause $Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond l \in T$ and $V' \models Q$. □

LEMMA 5.20. *Let T be an augmented set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses and $H = (N, E)$ be the labelled behaviour graph for T. Then for every node $n = (V, E_A, E_E)$ in H, if $l_{\langle ind\rangle} \in E_E$ and $V \not\models l$, then $V \models w^{ind}_l$.*

PROOF. The proof proceeds analogously to the proof of Lemma 5.18 and uses the fact that T contains the clause $w^{ind}_l \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(w^{ind}_l \vee l)$. □

LEMMA 5.21. *Let T be an augmented set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses and T' be a set of* $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ *clauses obtained from T by adding any combination of initial, **A**-step, **E**-step, or global clauses which only involve propositions and indices occurring in T. Then the labelled behaviour graph $H' = (N', E')$ for T' is a subgraph of the labelled behaviour graph $H = (N, E)$ of T.*

PROOF. This is established by induction on the length of the shortest path from an initial node to a node in H'. For the base case, where the length of the path is zero, we show that any initial node in H' is an initial node in H.

As T' has been constructed by adding a combination of initial, **A**-step, **E**-step, and global clauses to T, we have $R_0(T) \subseteq R_0(T')$ and $R_g(T) \subseteq R_g(T')$. Take any initial node $n_0 = (V^0, E^0_A, E^0_E)$ in H'. By Definition 5.12, V^0 satisfies $R_0(T') \cup R_g(T')$. As $R_0(T) \subseteq R_0(T')$ and $R_g(T) \subseteq R_g(T')$, then V^0 must also satisfy $R_0(T) \cup R_g(T)$. As the set of **A**- and **E**-sometime clauses in T and T' is the same, V^0 satisfies the left-hand side of the same **A**- and **E**-sometime clauses, and the sets E^0_A and E^0_E will be the same in both graphs. Therefore, n_0 is also an initial node in H.

Next we assume that every node $n_i = (V^i, E^i_A, E^i_E)$, where the length of the shortest path in H' from an initial node to n_i is m, is in H. We show that every node $n_{i+1} = (V^{i+1}, E^{i+1}_A, E^{i+1}_E)$ in H' with an incoming edge $(n_i, ind, n_{i+1}) \in E'$, $ind \in \mathrm{Ind}(T)$ is also in H.

V^{i+1} satisfies $R_g(T') \cup R_A(n_i, T') \cup R_{ind}(n_i, T')$. Thus V^{i+1} also satisfies $R_g(T) \cup R_A(n_i, T) \cup R_{ind}(n_i, T)$, as $R_g(T) \subseteq R_g(T')$, $R_A(n_i, T) \subseteq R_A(n_i, T')$, and $R_{ind}(n_i, T) \subseteq R_{ind}(n_i, T')$. Furthermore as T and T' contain the same **A**- or **E**-sometime clauses in T, E^{i+1}_A and E^{i+1}_E will be the same in both graphs. Thus n_{i+1} is also present in H as is the edge (n_i, ind, n_{i+1}).

The proof that all the edges in H' are also in H is analogous to the previous proof for nodes.

Therefore, $N' \subseteq N$, $E' \subseteq E$ and $H' \subseteq H$. □

Definition 5.22 (*Terminal Node*). A node n in a labelled behaviour graph for an augmented set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses is a terminal node iff there exists an index $ind \in \mathrm{Ind}(T)$ such that no edges labelled with ind depart from n.

Definition 5.23 (*Ind-Labelled Terminal Subgraph for $l_{\langle ind \rangle}$*). For a labelled behaviour graph (N, E) for an augmented set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, a subgraph (N', E') is an *ind*-labelled terminal subgraph for $l_{\langle ind \rangle}$ of (N, E) iff

(ITS1) $N' \subseteq N$ and $E' \subseteq E$;
(ITS2) For all nodes $n, n' \in N$ and edges $(n, ind', n') \in E$, $n' \in N'$ and $(n, ind', n') \in E'$ iff $n \in N'$ and $ind = ind'$; and
(ITS3) For every node $n = (V, E_A, E_E) \in N'$, $l_{\langle ind \rangle} \in E_E$ and $V \models \neg l$.

Definition 5.24 (*Terminal Subgraph for l*). For a labelled behaviour graph (N, E) for an augmented set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, a subgraph (N', E') is a terminal subgraph for l of (N, E) iff

(TS1) $N' \subseteq N$ and $E' \subseteq E$;
(TS2) For every node $n \in N'$, there exists some index $ind \in \mathrm{Ind}(T)$ such that for all edges $(n, ind, n') \in E$, $n' \in N'$ and $(n, ind, n') \in E'$; and
(TS3) For every node $n = (V, E_A, E_E) \in N'$, $l \in E_A$ and $V \models \neg l$.

LEMMA 5.25. *Given a labelled behaviour graph $H = (N, E)$ and a node $n = (V, E_A, E_E) \in N$, if, for every eventuality $l_{\langle ind \rangle} \in E_E$, $l_{\langle ind \rangle}$ can be satisfied in n or in some node ind-reachable from n, then n is not in any ind-labelled terminal subgraph $H' = (N', E')$ for $l_{\langle ind \rangle}$ of H.*

PROOF. Let $H' = (N', E')$ be an arbitrary *ind*-labelled terminal subgraph for some arbitrary eventuality $l_{\langle ind \rangle}$ of H. Proving this lemma is equivalent to proving that if $n \in N'$, then $l_{\langle ind \rangle}$ cannot be satisfied in n nor in any nodes *ind*-reachable from n in H. Assume that $n \in N'$. According to property (ITS2), all nodes which are *ind*-reachable from n are also in N'. By property (ITS3), for every node $n' = (V', E'_A, E'_E) \in N'$, $l_{\langle ind \rangle} \in E'_E$ and l is not satisfied in n'. Therefore, $l_{\langle ind \rangle}$ cannot be satisfied in n nor in any node *ind*-reachable from n in H. □

Definition 5.26 (*Reduced Labelled Behaviour Graph*). Given a labelled behaviour graph $H = (N, E)$ for an augmented set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses T, the *reduced labelled behaviour graph H_{red}* for T is the result of exhaustively applying the following *deletion rules* to H.

(1) If $n \in N$ is a terminal node with respect to an index in $\mathrm{Ind}(T)$, then delete n and every edge into or out of n.
(2) If there is an *ind*-labelled terminal graph (N', E') of H such that $ind \in \mathrm{Ind}(T)$, then delete every node $n \in N'$ and every edge into or out of nodes in N'.
(3) If there is a terminal graph (N', E') of H with respect to some indices in $\mathrm{Ind}(T)$, then delete every node $n \in N'$ and every edge into or out of nodes in N'.

LEMMA 5.27. *If an augmented set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses T is unsatisfiable, then its reduced labelled behaviour graph H is empty.*

PROOF. Proving this lemma is equivalent to proving that, if H is not empty, then T is satisfiable. By the definition of the satisfiability of a CTL formula, it is also equivalent to proving that if H is not empty, then a CTL model structure satisfying T can be constructed from H. Therefore, we assume that the reduced labelled behaviour graph $H = (N, E)$ of T is non-empty, and we show how to construct a CTL model structure $M = \langle S, R, L, [\_], s_0 \rangle$ satisfying T from H.

According to the definition of a CTL model structure and the semantics of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, the following properties are necessary and sufficient for M to satisfy T.

(P1) $L(s_0)$ must satisfy $R_0(T) \cup R_g(T)$.

(P2) Every pair $(s_i, s_{i+1}) \in R$ must satisfy the set of **A**-step, **E**-step, and global clauses in T, that is, as follows.
- —$L(s_i)$ and $L(s_{i+1})$ satisfy $R_g(T)$.
- —For every **A**-step clause $P \Rightarrow \mathbf{A}\bigcirc Q \in T$, if $L(s_i)$ satisfies P, then $L(s_{i+1})$ must satisfy Q.
- —For every **E**-step clause $P \Rightarrow \mathbf{E}\bigcirc Q_{\langle ind\rangle} \in T$, if $L(s_i)$ satisfies P and $(s_i, s_{i+1}) \in [ind]$, then $L(s_{i+1})$ must satisfy Q.

(P3) For every **E**-sometime clause $P \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond l \in T$ and every state $s \in S$, if $M, s \models P$, then the path $\chi_s^{\langle ind\rangle}$ must contain a state $s' \in S$ such that $l \in L(s')$.

(P4) For every **A**-sometime clause $P \Rightarrow \mathbf{A}\Diamond l \in T$ and every state $s \in S$, if $M, s \models P$, then every path χ_s must contain a state $s' \in S$ such that $l \in L(s')$.

Now we inductively define the construction of a CTL model structure from a reduced labelled behaviour graph $H = (N, E)$ and a mapping h from M to H.

Let cs be a function such that $cs(n)$, for every node $n = (V, E_A, E_E)$, is a fresh state s such that $L(s) = V$. In addition, by $RP(s_n)$ we denote a reverse path consisting of a finite sequence $s_n, s_{n-1}, \ldots, s_0$ of states such that $s_n, s_{n-1}, \ldots, s_0 \in S$, s_0 is the root of M, and for every i, $0 \le i \le n-1$, $(s_i, s_{i+1}) \in R$.

The state s_0 of M is given by $s_0 = cs(n_0)$, where n_0 is an arbitrary initial node in H, and we define $h(s_0) = n_0$. By the construction of H, property (P1) holds for s_0.

Suppose we have constructed the state s_i for M and $RP(s_i) = s_i, s_{i-1}, \ldots, s_0$. Then our task is to choose for each index $ind \in \mathrm{Ind}(T)$ a pair $(s_i, s_{i+1}) \in [ind]$ for M. Assume $h(s_i) = n$ and n has k ind-successors $(n_1, n_2, \ldots, n_k)$ ordered in an arbitrary but fixed order ($k > 0$ as otherwise n would be a terminal node in H). Let S^{RP} be the set $\{s_j \mid s_{j-1}, s_j \in RP(s_i), h(s_{j-1}) = n, h(s_j) \in \{n_1, n_2, \ldots, n_k\}$ and $(s_{j-1}, s_j) \in [ind]\}$.

- —if the set S^{RP} is empty, then $s_{i+1} = cs(n_1)$ and $h(s_{i+1}) = n_1$;
- —else, let $s \in S^{RP}$ be the state such that the distance between s_i and s is the shortest among all the distances between s_i and a state in S^{RP} and assume $h(s) = n_m \in \{n_1, n_2, \ldots, n_k\}$, $1 \le m \le k$, then
  - —$s_{i+1} = cs(n_{m+1})$ and $h(s_{i+1}) = n_{m+1}$, if $m \ne k$;
  - —$s_{i+1} = cs(n_1)$ and $h(s_{i+1}) = n_1$, if $m = k$.

By this algorithm, for an arbitrary path χ_{s_0}, if a node n is used infinitely often to construct states $s \in \chi_{s_0}$ and the index ind is used infinitely often to construct the successor states of s on χ_{s_0}, then ind-successors of the node n are fairly chosen to construct the path χ_{s_0}. This ensures that all eventualities are satisfied in M, as will be shown later.

Following the instructions we provided and using a breadth-first order for the construction from the state s_0, a CTL model structure M is constructed from H. By the construction of M and H, property (P2) holds for M.

Now we prove the model structure M we constructed satisfies property (P3). Assume the clause $P \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond l$ is in T and let s be an arbitrary state in S such that $M, s \models P$. We need to show that the path $\chi_s^{\langle ind\rangle}$ contains a state s' such that $l \in L(s')$. We give a proof by contradiction.

Assume l does not hold on $\chi_s^{\langle ind\rangle}$. We know the path $\chi_s^{\langle ind\rangle}$ is an infinite sequence, whereas the set of nodes in H is finite, which implies that there are nodes $\{n_1^t, n_2^t, \ldots, n_k^t\} \in H, k \ge 1$, that are used infinitely often to construct the path $\chi_s^{\langle ind\rangle}$.

As we assume that l does not hold on $\chi_s^{\langle ind \rangle}$, we obtain that, for every state $s'' \in \chi_s^{\langle ind \rangle}$, $M, s'' \not\models l$. Therefore, for every node $h(s'') = (V'', E''_A, E''_E)$, $V'' \not\models l$. Moreover, by the construction of H and M, $P \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc l \in T$ and $M, s \models P$, we obtain that $l_{\langle ind \rangle} \in E''_E$. Therefore, for $1 \leq i \leq k$, $n_i^t = (V_i^t, E_{A_i}^t, E_{E_i}^t)$, $V_i^t \not\models l$ and $l_{\langle ind \rangle} \in E_{E_i}^t$. By the way we construct M, all the *ind*-successors of each node in the set $\Delta = \{n_1^t, n_2^t, \dots, n_k^t\}$ are also in Δ. Thus, the set of nodes $\{n_1^t, n_2^t, \dots, n_k^t\}$ in H and all the *ind*-labelled edges departing from those nodes form an *ind*-labelled terminal subgraph for $l_{\langle ind \rangle}$ of H. However, H is a reduced labelled behaviour graph, so no *ind*-labelled terminal subgraph exists in H. We obtain a contradiction. Therefore, l must hold on the path $\chi_s^{\langle ind \rangle}$ and property (P3) holds for M.

The proof that property (P4) holds for M is analogous to the proof that property (P3) holds for M. □

LEMMA 5.28. *If a set of initial and global clauses is unsatisfiable, then there is a refutation using only step resolution rules.*

PROOF. If a set T of initial and global clauses is unsatisfiable, then the set $T' = \{D \mid \mathbf{true} \Rightarrow D \in T \text{ or } \mathbf{start} \Rightarrow D \in T\}$ is unsatisfiable by the semantics of $\mathbf{A}\Box$ and **start**.

The set T' only consists of propositional clauses. Therefore, it has a refutation by propositional ordered resolution with selection using the same ordering and selection function as for $\mathsf{R}_{\mathrm{CTL}}^{\succ,S}$. Then, we can use step resolution rules SRES4, SRES5, and SRES8 on this set T to derive a contradiction, namely, either **start** $\Rightarrow$ **false** or **true** $\Rightarrow$ **false**. □

LEMMA 5.29. *If the unreduced labelled behaviour graph for an augmented set of* $\mathrm{SNF}_{\mathrm{CTL}}^{\mathrm{g}}$ *clauses T is empty, then a contradiction can be obtained by applying step resolution rules to clauses in or derived from T.*

PROOF. If the unreduced labelled behaviour graph is empty, then by the definition of labelled behaviour graph, there are no initial nodes, which means there does not exist a valuation V such that the right-hand sides of all initial and global clauses of T are true under V. Thus, the subset of T containing all initial and global clauses in T is unsatisfiable, and by Lemma 5.28, there exists a refutation of T using step resolution rules SRES4, SRES5, and SRES8. □

THEOREM 5.30 (COMPLETENESS OF $\mathsf{R}_{\mathrm{CTL}}^{\succ,S}$). *If a finite augmented set T of* $\mathrm{SNF}_{\mathrm{CTL}}^{\mathrm{g}}$ *clauses is unsatisfiable, then T has a refutation using the resolution rules SRES1 to SRES8, ERES1, ERES2, and the rewrite rules RW1 and RW2.*

PROOF. Let T be an arbitrary augmented unsatisfiable set of $\mathrm{SNF}_{\mathrm{CTL}}^{\mathrm{g}}$ clauses and let $\succ$ and S be an arbitrary ordering and arbitrary selection function, respectively. The proof proceeds by induction on the sequence of applications of the deletion rules to the labelled behaviour graph of T. If the unreduced labelled behaviour graph is empty, then by Lemma 5.29, we can obtain a refutation by applying step resolution rules SRES4, SRES5, and SRES8.

Now suppose the labelled behaviour graph H is non-empty. The reduced labelled behaviour graph must be empty by Lemma 5.27, so there must be a node that can be deleted from H.

Suppose there is a node n which would be subject to the first deletion rule in Definition 5.26. So, there is an index $ind \in \mathrm{Ind}(T)$ such that n has no *ind*-successors. Then n is a terminal node $n = (V, E_A, E_E)$.

Let *Prop* be a function mapping determinate clauses to propositional clauses such that $\mathit{Prop}(P \Rightarrow *D) = D$ where $*$ is either empty or an operator in $\{\mathbf{A}\bigcirc, \mathbf{E}_{\langle ind \rangle}\bigcirc\}$. We extend *Prop* to a set N' of determinate clauses by $\mathit{Prop}(N') = \{\mathit{Prop}(C) \mid C \in N'\}$.

Let $W' = \{P \Rightarrow \mathbf{A}\bigcirc D \mid P \Rightarrow \mathbf{A}\bigcirc D \in T \text{ and } V \models P\} \cup \{P \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc D' \mid P' \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc D' \in T \text{ and } V \models P'\} \cup \{\mathbf{true} \Rightarrow D'' \mid \mathbf{true} \Rightarrow D'' \in T\}$, where P and P' are conjunctions of literals, whereas D, D', D'' are disjunctions of literals, and let $W = \mathit{Prop}(W')$. By Definition 5.12, W must be unsatisfiable, for otherwise, there would exist a node n' that could serve as an ind-successor of n.

Given that W is a set of propositional clauses, it has a refutation by propositional ordered resolution with selection using the same ordering and selection function as for $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$. In particular, let $N_0, N_1, \ldots, N_n$, where $N_0 = W$ and for every $i, 0 \leq i \leq n-1, N_{i+1} = N_i \cup \{C_{i+1}\}$ such that C_{i+1} is a propositional clause derived from N_i and $C_n = \mathbf{false}$, be a refutation of W.

We inductively construct a derivation $N'_0, N'_1, \ldots, N'_n$ from T by SRES1 to SRES3 and SRES6 to SRES8 such that for every i, $0 \leq i \leq n$, the following two properties hold for N'_i: (i) $\mathit{Prop}(N'_i) = N_i$ and (ii) for every determinate clause $P \Rightarrow *D$ in N'_i, where $*$ is either empty or an operator in $\{\mathbf{A}\bigcirc, \mathbf{E}_{\langle ind \rangle}\bigcirc\}$, $V \models P$. Then, by construction, N'_n contains a clause $\mathbf{true} \Rightarrow \mathbf{false}$, $P \Rightarrow \mathbf{A}\bigcirc\mathbf{false}$ or $P \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc\mathbf{false}$ with $ind \in \mathrm{Ind}(T)$, where P is a conjunction of literals and satisfied by V.

We prove the base case first. We define $N'_0 = W'$. Then by definition of W' and W, N'_0 satisfies properties (i) and (ii).

Next we prove the induction step. Assume that we have already constructed a derivation $N'_0, \ldots, N'_i$ such that for every j, $0 \leq j \leq i$, N'_j satisfies properties (i) and (ii). For the refutation $N_0, N_1, \ldots, N_i, N_{i+1}, \ldots, N_n$ of W, let $N_{i+1} = N_i \cup \{C_{i+1}\}$. We define N'_{i+1} by $N'_{i+1} = N'_i \cup \{C'_{i+1}\}$, where C'_{i+1} is the form of $\mathbf{true} \Rightarrow C_{i+1}$, $P_{i+1} \Rightarrow \mathbf{A}\bigcirc C_{i+1}$ with $V \models P_{i+1}$ or $P_{i+1} \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc C_{i+1}, ind \in \mathrm{Ind}(T)$ with $V \models P_{i+1}$ and derived by an application of one of the resolution rules SRES1 to SRES3 and SRES6 to SRES8 from N'_i.

Suppose $C_{i+1} = B_1 \vee B_2$ is derived from two clauses $B_1 \vee l$ and $B_2 \vee \neg l$, then by the construction of W we are able to find a clause $G = P_i \Rightarrow \mathbf{A}\bigcirc(B_1 \vee l)$, $P_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_1 \vee l)$ or $\mathbf{true} \Rightarrow B_1 \vee l$ and $G' = P'_i \Rightarrow \mathbf{A}\bigcirc(B_2 \vee \neg l)$, $P'_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_2 \vee \neg l)$ or $\mathbf{true} \Rightarrow B_2 \vee \neg l$ in N'_i. Note that if G and G' are both $\mathbf{E}$-step clauses, then the indices ind in them are identical. Depending on the form of G and G', we can distinguish the following cases.

$$\frac{\begin{array}{rl}\text{From } G = & P_i \Rightarrow \mathbf{A}\bigcirc(B_1 \vee l)\\ \text{and } G' = & P'_i \Rightarrow \mathbf{A}\bigcirc(B_2 \vee \neg l)\end{array}}{\text{we can derive } C'_{i+1} = P_i \wedge P'_i \Rightarrow \mathbf{A}\bigcirc(B_1 \vee B_2) \text{ by SRES1}},$$

$$\frac{\begin{array}{rl}\text{From } G = & P_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_1 \vee l)\\ \text{and } G' = & P'_i \Rightarrow \mathbf{A}\bigcirc(B_2 \vee \neg l)\end{array}}{\text{we can derive } C'_{i+1} = P_i \wedge P'_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_1 \vee B_2) \text{ by SRES2}},$$

$$\frac{\begin{array}{rl}\text{From } G = & P_i \Rightarrow \mathbf{A}\bigcirc(B_1 \vee l)\\ \text{and } G' = & P'_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_2 \vee \neg l)\end{array}}{\text{we can derive } C'_{i+1} = P_i \wedge P'_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_1 \vee B_2) \text{ by SRES2}},$$

$$\frac{\begin{array}{rl}\text{From } G = & P_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_1 \vee l)\\ \text{and } G' = & P'_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_2 \vee \neg l)\end{array}}{\text{we can derive } C'_{i+1} = P_i \wedge P'_i \Rightarrow \mathbf{E}_{\langle ind \rangle}\bigcirc(B_1 \vee B_2) \text{ by SRES3}},$$

$$\frac{\begin{array}{rl}\text{From } G = & \mathbf{true} \Rightarrow B_1 \vee l\\ \text{and } G' = & P'_i \Rightarrow \mathbf{A}\bigcirc(B_2 \vee \neg l)\end{array}}{\text{we can derive } C'_{i+1} = \quad P'_i \Rightarrow \mathbf{A}\bigcirc(B_1 \vee B_2) \text{ by SRES6}},$$

$$\frac{\begin{array}{rl}\text{From } G = & P_i \Rightarrow \mathbf{A}\bigcirc(B_1 \vee l)\\ \text{and } G' = & \mathbf{true} \Rightarrow B_2 \vee \neg l\end{array}}{\text{we can derive } C'_{i+1} = \quad P_i \Rightarrow \mathbf{A}\bigcirc(B_1 \vee B_2) \text{ by SRES6}},$$

$$\frac{\begin{array}{rl}\text{From } G = & \mathbf{true} \Rightarrow B_1 \vee l\\ \text{and } G' = & P'_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(B_2 \vee \neg l)\end{array}}{\text{we can derive } C'_{i+1} = \quad P'_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(B_1 \vee B_2) \text{ by SRES7}},$$

$$\frac{\begin{array}{rl}\text{From } G = & P_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(B_1 \vee l)\\ \text{and } G' = & \mathbf{true} \Rightarrow B_2 \vee \neg l\end{array}}{\text{we can derive } C'_{i+1} = \quad P_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc(B_1 \vee B_2) \text{ by SRES7}},$$

$$\frac{\begin{array}{rl}\text{From } G = & \mathbf{true} \Rightarrow B_1 \vee l\\ \text{and } G' = & \mathbf{true} \Rightarrow B_2 \vee \neg l\end{array}}{\text{we can derive } C'_{i+1} = \mathbf{true} \Rightarrow B_1 \vee B_2 \text{ by SRES8}},$$

where l is eligible in $B_1 \vee l$ and $\neg l$ is eligible in $B_2 \vee \neg l$ for a given atom ordering $\succ$ and a given selection function S of $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$, as otherwise, we would not have been able to derive $C_{i+1} = B_1 \vee B_2$ on the propositional level using ordered resolution with selection given the ordering $\succ$ and the selection function S.

Because $C_{i+1} = B_1 \vee B_2$, C'_{i+1} is one of the clauses $P_i \wedge P'_i \Rightarrow \mathbf{A}\bigcirc C_{i+1}$, $P_i \wedge P'_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc C_{i+1}$, $P_i \Rightarrow \mathbf{A}\bigcirc C_{i+1}$, $P'_i \Rightarrow \mathbf{A}\bigcirc C_{i+1}$, $P_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc C_{i+1}$, $P'_i \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc C_{i+1}$, or $\mathbf{true} \Rightarrow B_1 \vee B_2$. It is easy to see that since $V \models P_i$ and $V \models P'_i$, as $P_{i+1} = P_i \wedge P'_i$, we have $V \models P_{i+1}$. Furthermore, the preceding cases (SRES1 to SRES3 and SRES6 to SRES8) cover all the possibilities to derive C'_{i+1}. Thus, if there exists a derived clause C_{i+1}, then C'_{i+1} can be derived by $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$. Furthermore, by definition, N'_{i+1} satisfies properties (i) and (ii).

Thus, it follows that we can derive a clause $C'_n = P_n \Rightarrow \mathbf{A}\bigcirc\mathbf{false}$, $P_n \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\mathbf{false}$ or $\mathbf{true} \Rightarrow \mathbf{false}$ from T. From $P_n \Rightarrow \mathbf{A}\bigcirc\mathbf{false}$ or $P_n \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\mathbf{false}$, we can obtain the clause $\mathbf{true} \Rightarrow \neg P_n$ in normal form using RW1 or RW2.

By Lemma 5.21, the labelled behaviour graph H' for N'_n is a subgraph of H. In particular, every node in H' has to satisfy $\neg P_n$. Obviously, the node $n \in N$ does not satisfy this global clause and is thus not a node in H'.

Suppose the second (or third) deletion rule in Definition 5.26 is applicable to H. Then there must exist an eventuality $l_{\langle ind\rangle}$ (or l), where $l_{\langle ind\rangle}$ (or l) is not satisfied in an *ind*-labelled terminal subgraph for $l_{\langle ind\rangle}$ (or a terminal subgraph for l) of nodes *ind*-reachable (or reachable). We have two cases depending on the type of terminal subgraphs.

—*Ind-Labelled Terminal Subgraph for $l_{\langle ind\rangle}$*. Let $Q \Rightarrow \mathbf{E}_{\langle ind\rangle}\Diamond l$ be a clause in T and $H' = (N', E')$ be an *ind*-labelled terminal subgraph for $l_{\langle ind\rangle}$ of the behaviour graph H. For each $n = (V, E_A, E_E) \in N'$, let $loop(n)$ be the set consisting of all global, **A**-step, and **E**-step clauses labelled with *ind* in T whose left-hand sides are satisfied by V, and let $F_n \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc G_n$ be the clause, which is the result of merging all clauses in $loop(n) \cup \{\mathbf{true} \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\mathbf{true}\}$. To show the set $\bigcup_{n\in N'} loop(n)$ is an **E**-loop in $\neg l$, we must check the following two conditions.
  —For each $n \in N'$, we must have $\models G_n \Rightarrow \neg l$. In the following, we use a proof by contradiction to establish it. By the construction of H, G_n is the only constraint on the valuations of *ind*-successors of n. Therefore, if the implication $G_n \Rightarrow \neg l$ is not valid, then, by the construction of H, there must be an *ind*-successor

$n' = (V', E'_A, E'_E)$ of n, such that $V' \models l$. By property (ITS2), every *ind*-successor of n is in H'. By property (ITS3), for every *ind*-successor $n_i = (V^i, E^i_A, E^i_E)$ of the node n, $V^i \models \neg l$. Therefore, we obtain a contradiction. So, $G_n \Rightarrow \neg l$ is valid.

—For each $n \in N'$ we must have $\models G_n \Rightarrow \bigvee_{n' \in N'} F_{n'}$. Let $\{n_1, n_2, \ldots, n_k\}, k \geq 1$ be the set of *ind*-successors of the node n. We show that the assumption that $G_n \Rightarrow \bigvee_{n' \in N'} F_{n'}$ is not valid leads to a contradiction. By property (ITS2), every *ind*-successor of n is also in N'. Thus, to prove $\models G_n \Rightarrow \bigvee_{n' \in N'} F_{n'}$, it is enough to prove that $\models G_n \Rightarrow \bigvee_{i=1}^{k} F_{n_i}$. By the construction of H, G_n is the only constraint on the valuations of *ind*-successors of n. Therefore, if $G_n \Rightarrow \bigvee_{i=1}^{k} F_{n_i}$ is not valid, then, by the construction of H, there must be an *ind*-successor $n'' = (V'', E''_A, E''_E)$ of n, such that $V'' \models \neg(\bigvee_{i=1}^{k} F_{n_i})$, namely, $V'' \models \neg F_{n_1} \wedge \cdots \wedge \neg F_{n_k}$. As we know, for every *ind*-successor $n_i = (V^i, E^i_A, E^i_E)$, $1 \leq i \leq k$, of the node n, $V^{n_i} \models F_{n_i}$. Therefore, n'' can not be in the set $\{n_1, \ldots, n_k\}$. This is a contradiction. Thus, $G_n \Rightarrow \bigvee_{i=1}^{k} F_{n_i}$ is valid.

Since we show that the set $\bigcup_{n \in N'} loop(n)$ is an **E**-loop in $\neg l$, we are able to use it in an application of ERES2 with the eventuality $l_{\langle ind \rangle}$ occurring in $Q \Rightarrow \mathbf{E}_{\langle ind \rangle} \Diamond l \in T$. Let L be defined as

$$L = \bigvee_{n \in N'} F_n.$$

As L is a disjunction of conjunctions, in what follows, we use $w_l^{ind} \Rightarrow \mathbf{E}_{\langle ind \rangle} \bigcirc (\neg L \vee l)$ and **true** $\Rightarrow \neg Q \vee \neg L \vee l$ as a shorthand to denote the set of clauses corresponding to these formulae as specified in the ERSS2 rule given in Section 4.2. Then $T' = T \cup \{w_l^{ind} \Rightarrow \mathbf{E}_{\langle ind \rangle} \bigcirc (\neg L \vee l),\ \mathbf{true} \Rightarrow \neg Q \vee \neg L \vee l\}$ is the result of adding the resolvents derived by ERES2 to T. Note that, for every node $n = (V, E_A, E_E)$ in H', (i) $V \models L$; and (ii) by property (ITS3), $V \models \neg l$ and $l_{ind} \in E_E$. Therefore, $V \models \neg(\neg L \vee l)$. Moreover, by Lemma 5.20, $V \models w_l^{ind}$. Recall that through augmentation, the set T contains clauses

$$\begin{aligned} w_l^{ind} &\Rightarrow \mathbf{E}_{\langle ind \rangle} \bigcirc \big(l \vee w_l^{ind}\big), \\ \mathbf{true} &\Rightarrow \big(\neg Q \vee l \vee w_l^{ind}\big). \end{aligned}$$

By Lemma 5.19, either **(1)** there is an edge $(n', ind, n) \in E$, where $n' = (V', E'_A, E'_E)$, $l_{\langle ind \rangle} \in E'_E$, and $V' \models \neg l$; or **(2)** $V \models Q$, $V \models \neg l$.

(1) In the case of (1), we have $V' \models w_l^{ind}$ by Lemma 5.20. So, for the aforementioned resolvent $w_l^{ind} \Rightarrow \mathbf{E}_{\langle ind \rangle} \bigcirc (\neg L \vee l)$, V' satisfies w_l^{ind} but V does not satisfy $(\neg L \vee l)$. Thus, the labelled behaviour graph for T' does not contain an edge (n', ind, n). By the construction of the labelled behaviour graph for T', n is not a node in it.

(2) In the case of (2), we have $V \models Q$, $V \models L$, $V \models \neg l$. Thus, n does not satisfy the aforementioned resolvent **true** $\Rightarrow \neg Q \vee \neg L \vee l$ in T' and so n is not a node in the labelled behaviour graph for T'.

Therefore, the labelled behaviour graph for T' is a strict subgraph of that for T, and by induction, we assume that as T' has a refutation so must T.

—*Terminal subgraph for l*. The proof is analogous to the proof for *ind*-labelled terminal subgraphs for $l_{\langle ind \rangle}$. $\Box$

The behaviour graph construction and the use of deletion rules to remove parts of the behaviour graph which cannot contribute to the model structure for a given clause set is commonly used in completeness proofs for temporal and modal resolution calculi [Fisher et al. 2001] and resembles the tableau construction and marking procedure for propositional dynamic logic PDL in Pratt [1980].

5.3. Termination

In this section, we show that all the derivations obtained by applying rules of the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ to an arbitrary finite set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses terminate.

THEOREM 5.31. *Any derivation from a finite set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses by the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ terminates.*

PROOF. Assume T is augmented. Let T be constructed from a set Θ of n atomic propositions and a set Ind of m indices. Then the number of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses constructed from Θ and Ind is finite. We can have at most 2^{2n} initial clauses, 2^{2n} global clauses, 2^{4n} **A**-step clauses, $m \cdot 2^{4n}$ **E**-step clauses, $n \cdot 2^{2n+1}$ **A**-sometime clauses, and $m \cdot n \cdot 2^{2n+1}$ **E**-sometime clauses. In total, there can be at most $(m+1)2^{4n} + (m \cdot n + n + 1)2^{2n+1}$ different $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses. Any derivation from a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses by the calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ will terminate when either no more new clauses can be derived or a contradiction is obtained. Since there are only a finitely bounded number of different $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses, one of these two conditions will eventually be true. □

6. COMPLEXITY

The satisfiability problem of CTL is known to be EXPTIME-complete [Clarke and Emerson 1982; Emerson 1990; Emerson and Halpern 1985]. Next we consider the complexity of the decision procedure based on $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ and presented in Section 4.4.

THEOREM 6.1. *The complexity of the $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$-based decision procedure is in EXPTIME.*

PROOF. Assume that the set $\mathtt{N}$ of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses is augmented and is constructed from a set Θ of n propositions and a set Ind of m indices. The cost of deciding whether a step resolution rule can be applied to two determinate clauses is $\mathsf{A} = 4n+1$ in the worst case, provided we can compute $S(C)$ in linear time, compare literals in constant time, and check the identity of indices in constant time. From the proof of Theorem 5.31, we know the number of determinate clauses is at most $\mathsf{B} = 2^{2n}+2^{2n}+2^{4n}+m\cdot 2^{4n}$. Therefore, to naively compute a new clause from an application of some step resolution rule, we might need to look at $\mathsf{C} = \frac{\mathsf{B}(\mathsf{B}-1)}{2}$ combinations of two clauses, and the associated cost is $(\mathsf{C} \cdot \mathsf{A})$. Moreover, to decide whether the resolvent is a new clause or not, we need to compare the resolvent with at most B clauses and the cost is $\mathsf{D} = \mathsf{B} \cdot (8n^2 + 1)$. In the worst case, where each pair of clauses generates a resolvent but the resolvent already exists and only the last pair of clauses gives a new clause, to gain a new clause from an application of some step resolution rule, the complexity is of the order $(\mathsf{C} \cdot \mathsf{A} \cdot \mathsf{D})$, that is, EXPTIME. According to the proof of Theorem 5.31, there can be at most different B determinate clauses. Therefore, the complexity of saturating a set of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses by step resolution is the order of $(\mathsf{C} \cdot \mathsf{A} \cdot \mathsf{D}) \cdot \mathsf{B}$. That is the complexity of *resolution_sres* (line 8) is in EXPTIME.

To compute a new clause from an application of some eventuality resolution rule, the complexity depends on the complexity of the so-called CTL loop search algorithm which computes premises for the eventuality resolution rules [Bolotov and Dixon 2000]. The CTL loop search algorithm is a variation of the PLTL loop search algorithm [Dixon 1998] which has been shown to be in EXPTIME and we can show that the complexity of the CTL loop search algorithm from Bolotov and Dixon [2000] is also in EXPTIME. Generally speaking, each iteration of the CTL loop search algorithm is a saturation of the clause set, which is in EXPTIME, and there may be an exponential number of iterations required. Therefore, the complexity of *resolution_eres* (line 12) is in EXPTIME. According to the proof of Theorem 5.31, there can be at most distinct $n \cdot 2^{2n+1}$

A-sometime clauses and $m \cdot n \cdot 2^{2n+1}$ **E**-sometime clauses. Thus, the number of iterations for the for-loop (line 11 to 16) is at most $(n \cdot 2^{2n+1} + m \cdot n \cdot 2^{2n+1})$.

As we know, the cost of deciding whether a clause exists in the set `Old` is D and the number of clauses in the set `New` at line 17 is at most B. Therefore, the cost of the operation `New` \ `Old` is D · B (line 17). Analogously we can obtain that the cost of the operation `New` ∪ `G` (line 14) is bounded by D · B.

Thus, the complexity of each iteration of the do-while-loop (line 8 to 18) is in EXPTIME. From the proof of Theorem 5.31, there can be at most B different determinate clauses. The number of iterations of the do-while-loop is at most B. Therefore, the complexity of the decision procedure is in EXPTIME. □

7. RELATED WORK

First we provide a comparison with a previous resolution calculus for CTL and then we consider other approaches to determine the satisfiability of CTL and related logics.

7.1. Comparison between $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ and the Previous Resolution Calculus

$\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ is based on the resolution calculus for CTL in Bolotov [2000]. Both follow the spirit of the resolution calculus for PLTL in Fisher et al. [2001] which translates to a normal form and applies initial, step and eventuality resolution rules. The use of indices to transform CTL formulae into Separated Normal Form for CTL was introduced in Bolotov [2000]. However, no formal interpretation was given for indices, and no formal semantics stated for $\mathrm{SNF}^{g}_{\mathrm{CTL}}$. In this article, we provide a formal semantics for $\mathrm{SNF}^{g}_{\mathrm{CTL}}$.

Compared to the definition of $\mathrm{SNF}_{\mathrm{CTL}}$ in Bolotov [2000], we use an additional type of clause, namely, *global clauses*. Our definition of $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ provides several advantages over Bolotov [2000]. Global clauses inevitably occur as a result of inferences by step resolution rules. For example, from $m_1 \Rightarrow \mathbf{A}\bigcirc l$ and $m_2 \Rightarrow \mathbf{A}\bigcirc\neg l$, we can derive $m_1 \wedge m_2 \Rightarrow \mathbf{A}\bigcirc\mathbf{false}$, while from $m_1 \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc l$ and $m_2 \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\neg l$, we can derive $m_1 \wedge m_2 \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\mathbf{false}$. Both $m_1 \wedge m_2 \Rightarrow \mathbf{A}\bigcirc\mathbf{false}$ and $m_1 \wedge m_2 \Rightarrow \mathbf{E}_{\langle ind\rangle}\bigcirc\mathbf{false}$ are transformed into a global clause $\mathbf{true} \Rightarrow \neg m_1 \vee \neg m_2$ by RW1 and RW2, respectively.

As the normal form in Bolotov [2000] does not allow for such clauses, in the approach taken in Bolotov [2000], such global clauses must further be rewritten into equivalent pairs of an initial clause and an **A**-step clause as follows:

$$\mathbf{true} \Rightarrow \bigvee_{j=1}^{k} m_j \;\longrightarrow\; \begin{cases} \mathbf{start} \Rightarrow \bigvee_{j=1}^{k} m_j, \\ \mathbf{true} \;\Rightarrow \mathbf{A}\bigcirc \bigvee_{j=1}^{k} m_j, \end{cases}$$

where each m_j, $1 \leq j \leq k$, is a literal. For the same reason, in Bolotov [2000], the rewrite rules RW1 and RW2 will each produce two clauses, whereas in $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$, the analogous rewrite rules produce only one. Thus, one obvious advantage of allowing global clauses is that compared to Bolotov [2000], we will have fewer clauses transformed from the original CTL formula and generated by resolution.

In Bolotov [2000], a set of rules for the transformation of CTL formulae into a clausal normal form is provided. The transformation rules are shown to preserve satisfiability. However the proof for termination of the transformation process and the proof that the result of the process is indeed in normal form are absent and the complexity of the process is not studied. In contrast, we have provided the corresponding proofs and an analysis of the complexity for our transformation process.

Another difference to Bolotov [2000] is the approach taken in our completeness proof. The proof in Bolotov [2000] tries to relate the application of deletion rules on a CTL tableau to a sequence of resolution steps. In contrast, to show completeness of our

$$\textbf{TRES1}\quad \frac{\begin{array}{l}P^{\dagger} \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\square l\\ q \Rightarrow \mathbf{A}\diamond\neg l\end{array}}{q \Rightarrow \mathbf{A}(\neg P^{\dagger}\,\mathcal{W}\,\neg l)} \qquad \textbf{TRES2}\quad \frac{\begin{array}{l}P^{\dagger} \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\square l\\ q \Rightarrow \mathbf{E}_{\langle ind\rangle}\diamond\neg l\end{array}}{q \Rightarrow \mathbf{E}_{\langle ind\rangle}(\neg P^{\dagger}\,\mathcal{U}\,\neg l)}$$

where $P^{\dagger}$ is a disjunction of conjunctions of literals and l and q are literals.

Fig. 2. Redundant eventuality resolution rules.

calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ we construct a graph known as a *labelled behaviour graph*. This is an extension of the concept of a behaviour graph used in Fisher et al. [2001] for proving completeness of a clausal resolution for PLTL and related to the concept of a labelled behaviour graph used [Dixon et al. 2002]. However, our labelled behaviour graph differs in its construction to capture the semantics of indices in $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$. We believe our completeness proof demonstrates a closer relationship between the application of resolution rules and deletions in the labelled behaviour graph. Moreover, it is relatively easy to generate a CTL model structure from a non-empty reduced labelled behaviour graph and the labelled behaviour graph for a set T of $\mathrm{SNF}^{\mathrm{g}}_{\mathrm{CTL}}$ clauses saturated under $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ that contains no contradiction, is a reduced labelled behaviour graph. Hence, we could potentially use the labelled behaviour graph construction to generate counter models for failed proofs.

Furthermore, in the resolution calculus for CTL presented in Bolotov [2000] and Dixon et al. [2002], step resolution is not constrained by an ordering and a selection function. Therefore, their step resolution rules [Bolotov 2000; Dixon et al. 2002] allow for considerably more, and superfluous, inferences. In addition, this earlier resolution calculus contains four eventuality resolution rules, TRES1 to TRES4, where ERES1 and ERES2 correspond to TRES3 and TRES4, respectively. The other two eventuality resolution rules are given in Figure 2. Using our completeness proof, we can prove that the two eventuality resolution rules TRES1 and TRES2 in Bolotov [2000] and Dixon et al. [2002] are redundant.

We give a brief explanation why this is the case. Informally, the only difference between TRES1 and ERES1 is their first premise. For TRES1, it is $P^{\dagger} \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\square l$ and for ERES1, it is $P^{\dagger} \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\square l$. In Bolotov [2000] and Dixon et al. [2002], $\mathbf{A}\bigcirc\mathbf{A}\square l$ is called an **A**-loop and $\mathbf{E}\bigcirc\mathbf{E}\square l$ is called an **E**-loop. According to the semantics of CTL, $\mathbf{A}\bigcirc\mathbf{A}\square l \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\square l$, meaning if there exists an **A**-loop, there must be an **E**-loop as well. So, whenever we can apply TRES1 (TRES2), ERES1 (ERES2) is applicable as well. More formally, in our completeness proof, we only identify two types of subgraphs where some eventuality can not be fulfilled, namely, *ind*-labelled terminal subgraphs and terminal subgraphs. Both are **E**-loops according to the definition in Bolotov [2000] and the deletion of both types of subgraphs correlates to applications of ERES1 or ERES2. Thus, no further inference rules are required showing that TRES1 and TRES2 are redundant. Considering that the eventuality resolution rules are computationally very expensive, we gain a significant improvement here.

Finally, complexity of the method is not discussed in Bolotov [2000]. In this article, we prove that a decision procedure based on $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ is of the order EXPTIME.

7.2. Other Approaches for the Satisfiability Problem of CTL and Related Logics

There are other approaches, which can also be used to solve the satisfiability problem of CTL, namely, automata techniques [Vardi and Wolper 1986], Hintikka-based approaches [Emerson and Halpern 1985; Emerson 1990], and tableau calculi [Emerson and Halpern 1985; Emerson 1990; Abate et al. 2007; Attie 2003]. We give a brief discussion of these approaches.

The automata-based decision procedure for CTL [Vardi and Wolper 1986] consists of two separate phases. First, the decision procedure constructs an automaton A for a given CTL formula φ. This constructed automaton has a very useful property that it accepts certain infinite tree models of φ iff the formula φ is satisfiable. In the second phase the decision procedure checks whether there exists a tree model accepted by A, that is, whether the language accepted by A is non-empty. If there is one tree model accepted by A, then φ is satisfiable. Otherwise, φ is unsatisfiable. Constructing the automaton requires exponential time in the size of φ while checking the emptiness of the language accepted by the automaton requires polynomial time in the size of A.

Hintikka-based approaches first generate a closure set of formulae that may occur in the construction. All the possible states and edges are generated according a set of constraints, and some of these are deleted according to rules relating to the semantics of formulae in the states. This is analogous to our behaviour graph construction. For the former, the Fischer-Ladner closure for CTL formulae is defined. This defines all formulae that may appear in the construction. For example, if $\mathbf{E}\Diamond\varphi$ is in the closure, then both φ and $\mathbf{E}\bigcirc\mathbf{E}\Diamond\varphi$ are also in the closure. Additionally a list of constraints are provided on formulae that appear in state labels. For example, if $\mathbf{E}\Diamond\varphi$ is in a state label, then either φ or $\mathbf{E}\bigcirc\mathbf{E}\Diamond\varphi$ must also be in the state label. Initially all possible states and edges between them (with respect to formulae of the form $\mathbf{A}\bigcirc\varphi$) are constructed. States are removed if they do not satisfy the list of constraints and if they do not satisfy $\mathbf{E}\bigcirc$ formulae or eventualities (i.e., formulae for the form $\mathbf{P}\Diamond\varphi$ or $\mathbf{P}(\psi\,\mathcal{U}\,\varphi)$, where $\mathbf{P}$ is either path operator). The downside of this approach is that as all the states are constructed at the start the worst case complexity is always obtained.

With tableau calculi, the idea is to use the formula to be shown satisfiable to construct a structure such that a model can be constructed from that structure. Usually two types of rules are defined for the calculus, namely, the construction rules and the deletion rules. The construction rules, are used to expand an arbitrary CTL formula φ into a (possibly cyclic) graph such that each node of this graph is a set of CTL formulae. During the construction phase, typically tableau calculi utilise alpha and beta rules dependent on whether formulae are conjunctive or disjunctive. For example, $\varphi \wedge \psi$ is an alpha formula, where $\alpha_1 = \varphi$ and $\alpha_2 = \psi$, whereas $\varphi \vee \psi$ is an beta formula, where $\beta_1 = \varphi$ and $\beta_2 = \psi$. Temporal formulae are unwound to equivalent formulae relating to the current moment and the next moment, for example, $\mathbf{A}\Box\varphi \equiv \varphi \wedge \mathbf{A}\bigcirc\mathbf{A}\Box\varphi$ and $\mathbf{E}\Diamond\varphi \equiv \varphi \vee \mathbf{E}\bigcirc\mathbf{E}\Diamond\varphi$, where $\mathbf{A}\Box\varphi$ is an alpha formula with $\alpha_1 = \varphi$ and $\alpha_2 = \mathbf{A}\bigcirc\mathbf{A}\Box\varphi$ and $\mathbf{E}\Diamond\varphi$ is a beta formula with $\beta_1 = \varphi$ and $\beta_2 = \mathbf{E}\bigcirc\mathbf{E}\Diamond\varphi$. The structure is expanded by selecting a formula to be expanded. If the formula is an alpha formula, a single successor is constructed with the formula replaced by α_1 and α_2. For beta formulae, two successors are created one with the formula replaced by β_1 and one with it replaced by β_2. The structure is expanded until all no further alpha and beta rules can be applied, that is, formulae are either literals or formulae beginning with $\mathbf{A}\bigcirc$ or $\mathbf{E}\bigcirc$ operators. Edges to a set of nodes are made, one for each $\mathbf{E}\bigcirc$ operator containing φ for every formula $\mathbf{A}\bigcirc\varphi$ and ψ for some $\mathbf{E}\bigcirc\psi$.

The deletion rules of a tableau calculus for CTL determine which nodes should be removed from the constructed tableau. A node n is eliminated if any one of the following condition is satisfied: (i) there are inconsistencies in the node, for example, p and $\neg p$; (ii) the node has certain requirements on its successors and those requirements are not fulfilled by its successors; or (iii) the node contains unrealised eventualities, for example, $\mathbf{E}\Diamond p$ (or $\mathbf{A}(p\,\mathcal{U}\,q)$) in the node n and can not be realised in any (resp. all) path reachable from the node n. The deletion rules remove all the nodes which are not able to be used to construct a CTL model satisfying the CTL formula φ. If the tableau has no nodes left after the deletion process, the tableau is a *closed tableau* for CTL and the

CTL formula φ is unsatisfiable, otherwise the tableau is an *open tableau* for CTL and φ is satisfiable.

Emerson and Halpern [1985] and Emerson [1990] define both Hintikka and tableau algorithms for checking the satisfiability of CTL formulae in exponential time. A complete axiomatisation for CTL is also provided. The Hintikka approaches follow the preceding outline, first constructing all the states and then applying deletions. Additionally, a tableau algorithm that does not require all the states constructing initially is described using the two-phase construction and deletion rules along the lines just described.

In Marrero [2005], an algorithm to check satisfiability of CTL formulae is proposed that is based on both the Hintikka approach from Emerson and Halpern [1982, 1985] and the use of ordinary binary decision diagrams (OBDDs), a technique used in model checking. Rather than labelling states with formulae in a particular closure set, a subset of this is defined using propositions and formulae with $\mathbf{E}\bigcirc$ as the main connective. These are used as the set of boolean state variables. An identical set of primed variables are used to denote variables in successor states. The states and relations in the structure are also represented as quantified boolean formulae showing conditions relating to successor states. The fulfilment of eventualities are represented by fixpoints and computed iteratively. This has been implemented and run on a number of examples. However, Goré et al. [2011] notes flaws in the implementation and the prover does not seem to be publicly available.

A two-pass tableau algorithm which first constructs a structure and then carries out deletions for the branching-time logic $\mathcal{UB}$ is provided in Ben-Ari et al. [1981]. $\mathcal{UB}$ is a sublogic of CTL which does not contain the operators $\mathbf{A}\mathcal{U}$ and $\mathbf{E}\mathcal{U}$. Goré et al. [2011] have implemented this tableau algorithm, extended to allow for until operators as part of their evaluation of CTL provers.

Attie [2003] is primarily interested in synthesising a concurrent program from a specification. A variant of CTL is used as the language for specification. In the models, the relation R between states is labelled by an index relating to the number of processes, I. In the syntax of CTL, this is reflected by using an indexed some path next-time operator $\mathbf{E}\bigcirc_i\varphi$ (and its all paths equivalent). Whilst R is serial, that is, for all $s \in S$ $\exists s' \in S, i \in I$ such that $(s, i, s') \in R$, there might not be a successor for each member of I. Additionally the usual $\mathbf{E}\bigcirc$ operator is defined as the disjunction of the indexed some path operators. Note that the operators such as $\mathbf{E}\Diamond$ and $\mathbf{E}\mathcal{U}$ do not have indexed counterparts. There seems to be some relationship to the indices we use in this article. Here, indices are used for technical reasons (one for each some path operator) and are not allowed in the input formula, whereas in Attie [2003], indices are there to capture the specification of a particular process. Further, with our use of indices in the extended model structures, there is exactly one successor for each index which is not the case in Attie [2003]. Also here operators, such as $\mathbf{E}\Diamond$ and $\mathbf{E}\mathcal{U}$, are provided with an indexed semantics, unlike Attie [2003], where indices are only allowed with next operators. The tableau algorithm provided in the paper uses a construction/deletion style algorithm along the lines just described.

A one-pass tableau calculus for CTL is defined in Abate et al. [2007]. This is an extension of the one pass tableau algorithm for PLTL in Schwendimann [1998]. Rather than having a two-phase algorithm with a construction phase followed by a deletion phase, additional information is added to the tableau nodes to make such decisions locally. The tableau is explored in a depth-first, branch-by-branch manner. In the one-pass tableau for CTL, as well as a set of CTL formulae, nodes contain details of the history of the branch, whether the node is closed, and information about the satisfaction of eventualities. The history is determined by the parent of the node, whereas the closure and eventuality information is determined by the node's children. However, the one-pass tableau calculus results in a double-EXPTIME decision procedure [Abate et al. 2007]. This is higher than the complexity of CTL-RP, which is EXPTIME.

The one pass tableau algorithm for CTL is implemented as a built-in module in the Tableau Workbench (TWB) [Abate and Goré 2003]. The TWB is a generic framework for building automated theorem provers for arbitrary propositional logics which provides a general architecture and a high-level language which allows users to specify tableau rules and provers based on these rules. It provides a number of predefined provers for a wide range of logics, for example, propositional logic, linear-time temporal logic, and CTL. However, the aim of TWB was extensibility rather than efficiency. An optimised version of the one-pass tableau, called TreeTab, has also been implemented and used as part of the comparisons in Goré et al. [2011].

A popular alternative to deductive methods for checking the satisfiability of CTL is using model checking. Model checkers take a model of the system, usually some form of state transition system, and a formula to be checked on that model and check whether the formula is satisfied on all models from an initial state in the model. In Clarke et al. [1986], a polynomial-time model checking algorithm for properties specified in CTL is given. The algorithm labels the states of the model with subformulae of the property to be checked that hold there. Implementations of temporal logic model checkers have been developed, for example, NuSMV [Cimatti et al. 2002] allows CTL properties and has been applied to many systems.

CTL* [Emerson and Halpern 1986] is a branching-time temporal logic that includes both CTL and PLTL as sublogics. The satisfiability problem for CTL* is 2EXPTIME-complete [Vardi and Stockmeyer 1985; Emerson and Jutla 1988] and for model-checking is PSPACE [Clarke et al. 1986]. Reynolds has proposed a tableau calculus for CTL* in Reynolds [2011]. The logic RoCTL* [French et al. 2007] is a branching-time temporal logic with additional operators to deal with obligation and robustness. In McCabe-Dansted and Dixon [2010], CTL-like restrictions of RoCTL* are proposed, some of which can be translated into CTL to provide EXPTIME decision procedures for these fragments. Additionally a tableau calculus for a bundled version of the logic has been proposed [McCabe-Dansted 2008].

MLSolver [Friedmann and Lange 2009] is a prover for modal fixpoint logics. The implementation supports both the linear and modal μ-calculi, CTL* (and therefore its sublogics PLTL and CTL) and Propositional Dynamic Logic. The approach is to construct both a tableau and a deterministic automaton and take the product of these to obtain a parity game. This is then solved. The prover is used as part of the comparisons of CTL provers in Goré et al. [2011], but it should be noted that it is applicable to a wider class of logics than the other provers compared.

8. CONCLUSIONS

CTL [Emerson 1990] was introduced by Emerson et al. in the 1980s and now is a well-known branching-time temporal logic for the specification and verification of computational systems. Approaches to the satisfiability problem in CTL include automata techniques [Vardi and Wolper 1986], tableau calculi [Abate et al. 2007; Emerson and Halpern 1985], and a resolution calculus [Bolotov 2000].

The calculus for CTL in Bolotov [2000] is based on the ideas underlying a resolution calculus for PLTL [Fisher et al. 2001]. Here, we have provided a refined clausal resolution calculus $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$ for CTL. Compared with Bolotov [2000], we use an ordering and a selection function to restrict the applicability of step resolution rules and we have fewer eventuality resolution rules. We present a new completeness proof based on labelled behaviour graphs. Our completeness proof demonstrates a closer relationship between applications of resolution rules and deletions on labelled behaviour graphs. The proof shows that the additional eventuality resolution rules in Bolotov [2000], which are the most costly rules, are redundant. In addition, we prove that the complexity of a $\mathsf{R}^{\succ,S}_{\mathrm{CTL}}$-based decision procedure for CTL is EXPTIME. An implementation of this calculus has been shown to be competitive with other CTL provers, demonstrating "robustness" in that it tends to succeed eventually rather than failing (or succeeding) spectacularly.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their helpful comments.

REFERENCES

P. Abate and R. Goré. 2003. The Tableaux Work Bench. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'03)*. Lecture Notes in Computer Science, vol. 2796, Springer-Verlag, Berlin, 230–236.

P. Abate, R. Goré, and F. Widmann. 2007. One-pass Tableaux for computation tree logic. In *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*. Lecture Notes in Computer Science, vol. 4790, Springer-Verlag, Berlin, 32–46.

L. Afanasiev, M. Franceschet, M. Marx, and M. De Rijke. 2004. CTL model checking for processing simple XPath queries. In *Proceedings of the 11th International Symposium on Temporal Representation and Reasoning (TIME'04)*. 117–124.

P. C. Attie. 2003. On the implementation complexity of specifications of concurrent programs. In *Proceedings of the 17th International Conference on Distributed Computing (DISC'03)*. Lecture Notes in Computer Science, vol. 2848, Springer-Verlag, Berlin, 151–165.

L. Bachmair and H. Ganzinger. 2001. Resolution theorem proving. In *Handbook of Automated Reasoning*. vol. 1, Elsevier, 19–99.

A. Basso and A. Bolotov. 2007. Towards GCM re-configuration extending specification by norms. In *Making Grids Work: Proceedings of the CoreGRID Workshop on Programming Models Grid and P2P System Architecture Grid Systems, Tools and Environments*. 17–29.

M. Ben-Ari, Z. Manna, and A. Pnueli. 1981. The temporal logic of branching time. In *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'81)*. ACM Press, 164–176.

A. Bolotov. 2000. Clausal resolution for branching-time temporal logic. Ph.D. dissertation, Manchester Metropolitan University.

A. Bolotov and C. Dixon. 2000. Resolution for branching time temporal logics: Applying the temporal resolution rule. In *Proceedings of the 7th International Workshop on Temporal Representation and Reasoning (TIME'00)*. 163–172.

A. Bolotov and M. Fisher. 1999. A clausal resolution method for CTL branching-time temporal logic. *J. Exp. Theor. Artif. Intell.* 11, 1, 77–93.

A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. 2002. NuSMV 2: An OpenSource tool for symbolic model checking. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV'02)*. Lecture Notes in Computer Science, vol. 2404, Springer-Verlag, Berlin, 359–364.

E. M. Clarke and E. A. Emerson. 1982. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceedings of the Logic of Programs Workshop*. Lecture Notes in Computer Science, vol. 131, Springer-Verlag, Berlin, 52–71.

E. M. Clarke, E. A. Emerson, and A. P. Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* 8, 2, 244–263.

E. M. Clarke, O. Grumberg, and D. A. Peled. 2000. *Model Checking*. MIT Press.

C. Dixon. 1998. Temporal resolution using a breadth-first search algorithm. *Ann. Math. Artif. Intell.* 22, 1–2, 87–115.

C. Dixon, M. Fisher, and A. Bolotov. 2002. Clausal resolution in a logic of rational agency. *Artif. Intell.* 139, 1, 47–89.

E. A. Emerson. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, Elsevier, Chapter 16, 996–1072.

E. A. Emerson and J. Y. Halpern. 1982. Decision procedures and expressiveness in the temporal logic of branching time. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC'82)*. ACM Press, 169–180.

E. A. Emerson and J. Y. Halpern. 1985. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Syst. Sci.* 30, 1, 1–24.

E. A. Emerson and J. Y. Halpern. 1986. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. ACM* 33, 1, 151–178.

E. A. Emerson and C. S. Jutla. 1988. Complexity of tree automata and modal logics of programs. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS'88)*. 328–337.

M. C. Fernández Gago. 2004. Efficient control of temporal reasoning. Ph.D. dissertation, University of Liverpool.

M. Fisher, C. Dixon, and M. Peim. 2001. Clausal temporal resolution. *ACM Trans. Comput. Logic* 2, 1, 12–56.

T. French, J. Mccabe-Dansted, and M. Reynolds. 2007. A temporal logic of robustness. In *Proceedings of the 6th International Symposium on Frontiers of Combining Systems* (*FroCoS'07)*. Lecture Notes in Computer Science, vol. 4720, Springer-Verlag, Berlin, 193–205.

O. Friedmann and M. Lange. 2009. A solver for modal fixpoint logics. *Electron. Notes Theor. Comput. Sci.* 262, 99–111.

R. Goré, J. Thomson, and F. Widmann. 2011. An experimental comparison of theorem provers for CTL. In *Proceedings of the 18th International Symposium on Temporal Representation and Reasoning (TIME'11)*. 49–56.

M. Huth and M. Ryan. 2004. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press.

Z. Manna and A. Pnueli. 1992. *The Temporal Logic of Reactive and Concurrent Systems*. Springer.

W. Marrero. 2005. Using BDDs to decide CTL. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*. Lecture Notes in Computer Science, vol. 3440, Springer-Verlag, Berlin, 222–236.

J. McCabe-Dansted. 2008. A tableau for RoBCTL. In *Proceedings of the 11th European Conference on Logics in Artificial Intelligence (JELIA'08)*. Lecture Notes in Computer Science, vol. 5293, Springer-Verlag, Berlin, 298–310.

J. McCabe-Dansted and C. Dixon. 2010. CTL-like fragments of a temporal logic of robustness. In *Proceedings of the 17th International Symposium on Temporal Representation and Reasoning (TIME'10)*. 11–18.

V. R. Pratt. 1980. A near-optimal method for reasoning about action. *J. Comput. Syst. Sci.* 20, 2, 231–254.

M. Reynolds. 2011. A tableau-based decision procedure for CTL*. *Formal Aspects Comput.* 23, 6, 1–41.

J. A. Robinson. 1965. Automatic deduction with hyper-resolution. *Int. J. Comput. Math.* 1, 227–234.

S. Schwendimann. 1998. A new one-pass tableau calculus for PLTL. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*. Lecture Notes in Computer Science, vol. 1397, Springer-Verlag, Berlin, 277–291.

M. Vardi and L. Stockmeyer. 1985. Improved upper and lower bounds for modal logics of programs. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC'85)*. ACM Press, 240–251.

M. Y. Vardi and P. Wolper. 1986. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.* 32, 2, 183–221.

C. Weidenbach, R. A. Schmidt, T. Hillenbrand, R. Rusev, and D. Topic. 2007. System description: Spass version 3.0. In *Proceedings of the 21st International Conference on Automated Deduction (CADE-21)*. Lecture Notes in Computer Science, vol. 4603, Springer-Verlag, Berlin, 514–520.

L. Zhang. 2010. Clausal reasoning for branching-time logics. Ph.D. dissertation, University of Liverpool.

L. Zhang, U. Hustadt, and C. Dixon. 2009a. A refined resolution calculus for CTL. In *Proceedings of the 22nd International Conference on Automated Deduction (CADE-22)*. Lecture Notes in Computer Science, vol. 5663, Springer-Verlag, Berlin, 245–260.

L. Zhang, U. Hustadt, and C. Dixon. 2009b. CTL-RP: A computational tree logic resolution prover. *AI Commun.* 23, 2–3, 111–136.

Received October 2012; revised May 2013; accepted September 2013