Annals of Mathematics and Artificial Intelligence (2006) 46: 235–263
DOI: 10.1007/s10472-006-9018-1

A clausal resolution method for branching-time logic ECTL$^+$

Alexander Bolotov and Artie Basukoski

Harrow School of Computer Science, University of Westminster, Watford Road, HA1 3TP, UK

E-mail: {A.Bolotov; A.Basukoski}@wmin.ac.uk

We expand the applicability of the clausal resolution technique to the branching-time temporal logic ECTL$^+$. ECTL$^+$ is strictly more expressive than the basic computation tree logic CTL and its extension, ECTL, as it allows Boolean combinations of fairness and single temporal operators. We show how any ECTL$^+$ formula can be translated to a normal form the structure of which was initially defined for CTL and then used for ECTL. This enables us to apply to ECTL$^+$ a resolution technique defined over the set of clauses. Both correctness of the method and complexity of the transformation procedure are given.

Keywords: automated deduction, branching-time logic, program specification and verification, resolution

AMS subject classification: 03B35, 03B44, 68N30, 68Q60, 68Q85, 68T15

1. Introduction

CTL type branching-time temporal logics play a significant role in potential applications such as specification and verification of concurrent and distributed systems [9]. Two combinations of future time temporal operators $\diamondsuit$ ('sometime') and $\square$ ('always'), are useful in expressing *fairness* [8]: $\diamondsuit\square p$ (p is true along the path of the computation except possibly some finite initial interval of it) and $\square\diamondsuit p$ (p is true along the computation path at infinitely many moments of time). The logic ECTL (Extended CTL [11]) was defined to enable the use of these simple fairness constraints. The logic ECTL$^+$ further extends the expressiveness of ECTL by allowing Boolean combinations of elementary temporal operators and ECTL fairness constraints (but not permitting nesting of temporal operators or fairness constraints). In [3, 4] a clausal resolution method has been developed for the logic ECTL. The introduction of the corresponding technique to cope with fairness constraints enabled the translation of ECTL formulae into the normal form, to which we apply a clausal resolution technique initially defined for the logic CTL. In this paper we present the translation to the normal form for any ECTL$^+$ formula. Similarly to ECTL, as a normal form we utilise the Separated Normal Form developed for CTL formulae, called SNF$_{\mathrm{CTL}}$. This enables us to apply the resolution technique defined over SNF$_{\mathrm{CTL}}$ as the refutation technique for ECTL$^+$ formulae.

The main contribution of this paper is the formulation of the technique to translate ECTL$^+$ formulae into SNF$_{\mathrm{CTL}}$ and a proof of its correctness.

The structure of the paper is as follows. In section 2 we outline the syntax and semantics of ECTL$^+$ and in section 3 we recall those properties of ECTL$^+$ that are important for our analysis. In section 4 we review SNF$_{\mathrm{CTL}}$. Next, in section 5, we describe the main stages of the algorithm to translate an ECTL$^+$ formula into SNF$_{\mathrm{CTL}}$, give details of rules invoked in this algorithm and provide the example transformation. The proof of the correctness of this transformation technique is given in section 6 and the proof of its complexity in section 7. Further, in section 8 we outline the temporal resolution method defined over SNF$_{\mathrm{CTL}}$ and apply it to a set of SNF$_{\mathrm{CTL}}$ clauses (previously obtained in section 5.3). Finally, in section 9, we draw conclusions and discuss future work.

2. Syntax and semantics of ECTL$^+$

2.1. *Syntax of ECTL$^+$*

Since we utilize ECTL$^+$ for the purposes of formal specification and verification, we define its language based upon the extended set of classical logic operators $\wedge, \vee, \Rightarrow, \neg$, the set of future time temporal operators $\square$ (always), $\diamondsuit$ (sometime), $\bigcirc$ (next time), $\mathcal{U}$ (until) and $\mathcal{W}$ (unless) and path quantifiers **A** (on all future paths) and **E** (on some future path). This will also unify our presentation with the definition of the normal form (see section 4).

First, we fix a countable set, $Prop = x, y, z, \ldots$, of atomic propositions. In the syntax of ECTL$^+$, similar to CTL and ECTL, we distinguish *state* (S) and *path* (P) formulae, such that well formed formulae are state formulae. These are inductively defined below (where C is a formula of classical propositional logic)

$$\begin{array}{lll} S & ::= & C|S \wedge S|S \vee S|S \Rightarrow S|\neg\, S|\mathbf{A}P|\mathbf{E}P \\ P & ::= & P \wedge P|P \vee P|P \Rightarrow P|\neg\, P| \\ & & \square\, S|\diamondsuit\, S|\bigcirc\, S|S\;\mathcal{U}\;S|S\;\mathcal{W}\;S|\square\,\diamondsuit S|\diamondsuit\,\square\, S \end{array}$$

Examples of ECTL$^+$ formulae that are not expressible in a weaker logic, ECTL, are $\mathbf{A}(\diamondsuit\,\square\, p \wedge \square\diamondsuit p)$, $\mathbf{E}(\diamondsuit\,\square\, p \vee \bigcirc\,\neg p$ (where $p \in Prop$). These formulae express the Boolean combination of fairness properties or temporal operators in the scope of a path quantifier.

Note that a succinct representation of branching-time logics which invokes a minimum set of temporal logic operators, $\mathcal{U}$ and $\bigcirc$, (from which we can derive other operators), can be found, for example, in [8].

2.2. *Semantics of ECTL$^+$*

We precede the presentation of the ECTL$^+$ semantics by the introduction of notations of tree structures, the underlying structures of time assumed for the logic under consideration.

Definition 1 (Tree). A tree, $\mathcal{T}$, is a pair (S, R), where S is a set of states and $R \subseteq S \times S$ is a relation between states of S *such that*

- $s_0 \in S$ is a unique root node, i.e., there is no state $s_i \in S$ such that $R(s_i, s_0)$;
- for every $s_i \in S$ there exists $s_j \in S$ such that $R(s_i, s_j)$;
- for every $s_i, s_j, s_k \in S$, if $R(s_i, s_k)$ and $R(s_j, s_k)$ then $s_i = s_j$.

A *path*, χ_{s_i} is a sequence of states $s_i, s_{i+1}, s_{i+2} \ldots$ such that for all $j \geqslant i$, $(s_j, s_{j+1}) \in R$. A path χ_{s_0} is called a *fullpath*. Let X be a family of all fullpaths of $\mathcal{T}$. Given a path χ_{s_i} and a state $s_j \in \chi_{s_i}$, $(i < j)$ we term a finite subsequence $[s_i, s_j] = s_i, s_{i+1}, \ldots, s_j$ of χ_{s_i} *a prefix* of a path χ_{s_i} and an infinite sub-sequence $s_j, s_{j+1}, s_{j+2}, \ldots$ of χ_{s_i} *a suffix* of a path χ_{s_i} abbreviated $Suf(\chi_{s_i}, s_j)$.

Definition 2 (Countable ω-tree). A countable *ω-tree*, $\mathcal{T}_\omega$, is a tree (S, R) with the family of all fullpaths, X, which satisfies the following conditions:

- each fullpath $\chi \in X$ is isomorphic to natural numbers;
- every state $s_i \in S$ has a countable number of successors.

Definition 3 (Branching degree of a state). The number of immediate successors of a state $s_i \in S$ *in a tree* (S, R) is called the branching degree of s_i.

Now we are ready to define the semantics for ECTL$^+$. A well-formed ECTL$^+$ formula is interpreted in a structure $\mathcal{M} = \langle S, R, s_0, X, L\rangle$, where (S, R) is a countable ω tree with a root s_0, X is a set of all fullpaths and L is an interpretation function mapping atomic propositional symbols to truth values at each state and the following conditions are satisfied:

- X is R-generable ([8]), i.e., for every state $s_i \in S$, there exists $\chi_j \in X$ such that $s_i \in \chi_j$, and for every sequence $\chi_j = s_0, s_1, s_2, \ldots$, the following is true: $\chi_j \in X$ if, and only if, for every i, $R(s_i, s_{i+1})$;
- a tree (S, R) is of at most countable branching.

In figure 1 we define a relation '$\models$', which evaluates well-formed ECTL$^+$ formulae at a state s_i in a model $\mathcal{M}$.

Definition 4 (Satisfiability). A well-formed ECTL$^+$ formula, B, is satisfiable if, and only if, there exists a model $\mathcal{M}$ such that $\langle \mathcal{M}, s_0\rangle \models B$.

Definition 5 (Validity). A well-formed ECTL$^+$ formula, B, is valid if, and only if, it is satisfied in every possible model.

s1. $\langle \mathcal{M}, s_i\rangle \models p$	iff	$p \in L(s_i)$, for $p \in Prop$.
s2. $\langle \mathcal{M}, s_i\rangle \models \neg A$	iff	$\langle \mathcal{M}, s_i\rangle \not\models A$
s3. $\langle \mathcal{M}, s_i\rangle \models A \wedge B$	iff	$\langle \mathcal{M}, s_i\rangle \models A$ and $\langle \mathcal{M}, s_i\rangle \models B$
s4. $\langle \mathcal{M}, s_i\rangle \models A \vee B$	iff	$\langle \mathcal{M}, s_i\rangle \models A$ or $\langle \mathcal{M}, s_i\rangle \models B$
s5. $\langle \mathcal{M}, s_i\rangle \models A \Rightarrow B$	iff	$\langle \mathcal{M}, s_i\rangle \not\models A$ or $\langle \mathcal{M}, s_i\rangle \models B$
s6. $\langle \mathcal{M}, s_i\rangle \models \mathbf{A}B$	iff	for each χ_{s_i}, $\langle \mathcal{M}, \chi_{s_i}\rangle \models B$.
s7. $\langle \mathcal{M}, s_i\rangle \models \mathbf{E}B$	iff	there exists χ_{s_i} such that $\langle \mathcal{M}, \chi_{s_i}\rangle \models B$
p1. $\langle \mathcal{M}, \chi_{s_i}\rangle \models A$	iff	$\langle \mathcal{M}, s_i\rangle \models A$, for state formula A
p2. $\langle \mathcal{M}, \chi_{s_i}\rangle \models \Box B$	iff	for each $s_j \in \chi_{s_i}$, if $i \leq j$ then $\langle \mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$.
p3. $\langle \mathcal{M}, \chi_{s_i}\rangle \models \Diamond B$	iff	there exists $s_j \in \chi_{s_i}$ such that $i \leq j$ and $\langle \mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$.
p4. $\langle \mathcal{M}, \chi_{s_i}\rangle \models \bigcirc B$	iff	$\langle \mathcal{M}, Suf(\chi_{s_i}, s_{i+1})\rangle \models B$.
p5. $\langle \mathcal{M}, \chi_{s_i}\rangle \models A \mathcal{U} B$	iff	there exists $s_j \in \chi_{s_i}$ such that $i \leq j$ and $\langle \mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$ and for each $s_k \in \chi_{s_i}$, if $i \leq k < j$ then $\langle \mathcal{M}, Suf(\chi_{s_i}, s_k)\rangle \models A$.
p6. $\langle \mathcal{M}, \chi_{s_i}\rangle \models A \mathcal{W} B$	iff	$\langle \mathcal{M}, \chi_{s_i}\rangle \models \Box A$ or $\langle \mathcal{M}, \chi_{s_i}\rangle \models A \mathcal{U} B$

Figure 1. ECTL$^+$ semantics.

As an example let us consider an ECTL$^+$ formula

$$\neg\mathbf{E}(\Box \Diamond p \wedge \Diamond \Box \neg p) \tag{1}$$

To show that (1) is valid we establish that the following formula, the negation of (1)

$$\mathbf{E}(\Box \Diamond p \wedge \Diamond \Box \neg p) \tag{2}$$

is unsatisfiable. We will show this by refutation, i.e., assuming that there is a structure, $\mathcal{M}$, such that its root satisfies $\mathbf{E}(\Box \Diamond p \wedge \Diamond \Box \neg p)$ and deriving a contradiction from this assumption. The steps of the refutation are as follows.

(1) $\langle \mathcal{M}, s_0\rangle \models \mathbf{E}(\Box \Diamond p \wedge \Diamond \Box \neg p)$.

(2) There exists a fullpath χ_{s_0} such that $\langle \mathcal{M}, \chi_{s_0}\rangle \models \Box \Diamond p \wedge \Diamond \Box \neg p$, from (1) by s7 in the ECTL$^+$ semantics.

(3) $\langle \mathcal{M}, \varphi_{s_0}\rangle \models \Box \Diamond p \wedge \Diamond \Box \neg p$, from (2) fixing χ_{s_0} as φ_{s_0}.

(4) $\langle \mathcal{M}, \varphi_{s_0}\rangle \models \Box \Diamond p$, from (2) by s3 in the ECTL$^+$ semantics.

(5) $\langle \mathcal{M}, \varphi_{s_0}\rangle \models \Diamond \Box \neg p$, from (2), by s3 in the ECTL$^+$ semantics.

(6) There exists a state $s_i \in \varphi_{s_0}(0 \leqslant i)$ such that $\langle \mathcal{M}, Suf(\varphi_{s_0}, s_i)\rangle \models \Box \neg p$, from (5) by p3 in the ECTL$^+$ semantics.

(7) $\langle \mathcal{M}, Suf(\varphi_{s_0}, s_j)\rangle \models \Box \neg p$, from (6) fixing s_i as s_j.

(8) For every state $s_k \in Suf(\varphi_{s_0}, s_j)(k \geqslant j)$, $\langle \mathcal{M}, s_k \rangle \models \neg p$, from (7) by p2, p1 in the ECTL$^+$ semantics.

(9) For every $l \geqslant 0 \langle \mathcal{M}, Suf(\varphi_{s_0}, s_l)\rangle \models \Diamond p$, from (4) by p2 in the ECTL$^+$ semantics.

(10) $\langle \mathcal{M}, Suf(\varphi_{s_0}, s_j)\rangle \models \Diamond p$, from (9) letting $s_l = s_j$.

(11) There exists a state $s_m \in Suf(\varphi_{s_0}, s_j)(j \leqslant m)$ such that $\langle \mathcal{M}, s_m \rangle \models p$, from (10) by p3 in the ECTL$^+$ semantics.

(12) $\langle \mathcal{M}, s_n \rangle \models p$, from (11) fixing s_m as s_n.

(13) $\langle \mathcal{M}, s_n \rangle \models \neg p$, from (8) letting $s_k = s_n$.

(14) contradiction: (12) and (13).

It is straightforward from the ECTL$^+$ semantics that since (2) is unsatisfiable then so is a stronger formula

$$\mathbf{A}(\Box \Diamond p \wedge \Diamond \Box \neg p) \tag{3}$$

This formula will serve in our example of the transformation towards SNF$_{\text{CTL}}$ in section 5.3.

Closure properties of ECTL$^+$ models. When trees are considered as models for distributed systems, paths through a tree are viewed as computations. The natural requirements for such models would be suffix and fusion closures. Following [8], the former means that every suffix of a path is itself a path. The latter requires that a system, following the prefix of a computation γ, at any point $s_j \in \gamma$, is able to follow any computation π_{s_j} originating from s_j.

Finally, we might require that "if a system can follow a path arbitrarily long, then it can be followed forever" [8]. This corresponds to limit closure property, meaning that for any fullpath γ_{s_0} and any paths $\pi_{s_j}, \varphi_{s_k}, \ldots$ such that γ_{s_0} has the prefix $[s_0, s_j]$, π_{s_j} has the prefix $[s_j, s_k]$, φ_{s_k} has the prefix $[s_k, s_l]$, etc, and $0 < j < k < l$, the following holds (see figure 2): there exists an infinite path α_{s_0} that is a limit of the prefixes $[s_0, s_j], [s_j, s_k], [s_k, s_l], \ldots$.

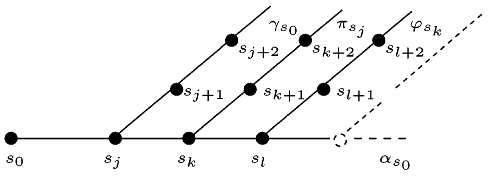

Figure 2. Limit closure.

In our definition of an ECTL$^+$ model structure $\mathcal{M}$ the set of fullpaths X is R-generable. Therefore, following [8], it satisfies all three closure properties, i.e., it is suffix, fusion and limit closed.

3. Some useful features of ECTL$^+$

Here we summarize those features of ECTL$^+$ that are important in our analysis and, thus, will affect both the translation of ECTL$^+$ formulae to the normal form and the clausal resolution method.

In the rest of the paper, let $\mathbf{T}$ abbreviate any unary and $\mathbf{T}^2$ any binary temporal operator and $\mathbf{P}$ either of path quantifiers. Any formula of the type $\mathbf{PT}$ or $\mathbf{PT}^2$ is called *a basic* CTL *modality*.

Proposition 1 (Negation Normal Form correctness). Given an ECTL$^+$ formula G and its Negation Normal Form $\mathrm{NNF}_{\mathrm{ECTL}^+}(G)$,

$$\langle \mathcal{M}, s_0 \rangle \models G \quad \text{iff} \quad \langle \mathcal{M}, s_0 \rangle \models \mathrm{NNF}_{\mathrm{ECTL}^+}(G) \ \ [8].$$

Given a CTL formula F, we will abbreviate the expression 'a state subformula F_i with a path quantifier as its main operator' by $\mathbf{P}$-*embedded* subformula of F. Now for an ECTL$^+$ formula F, we define a notion of the degree of nesting of its path quantifiers, denoted $N(F)$, as follows.

Definition 6 (Degree of path quantifier nesting).

- if F is a purely classical formula then $N(F) = 0$;
- if $F = \mathbf{T}F_1 | F_1\mathbf{T}^2F_2$, and F_1, F_2 are purely classical formulae then $N(\mathbf{T}F_1) = N(F_1\mathbf{T}^2F_2) = 0$;
- if $F = \neg F_1 | F_1 \wedge F_2 | F_1 \vee F_2 | F_1 \Rightarrow F_2 | \mathbf{T}F_1 | F_1\mathbf{T}^2F_2$ then $N(\neg F_1) = N(\mathbf{T}F_1) = N(F_1)$ and $N(F_1 \wedge F_2) = N(F_1 \vee F_2) = N(F_1 \Rightarrow F_2) = N(F_1\mathbf{T}^2F_2) = max(N(F_1), N(F_2))$;
- $N(\mathbf{P}F_1) = N(F_1) + 1$.

Emerson and Sistla [12] showed that by a continuous renaming of the $\mathbf{P}$-embedded state subformulae any CTL* (hence ECTL$^+$) formula F with $N(F) > 2$ can be transformed into F' such that $N(F') = 2$. For example, given $G = \mathbf{A}\Diamond(\mathbf{E}(\Box \Diamond \neg p \wedge \bigcirc q) \vee \mathbf{E}\Diamond\mathbf{E} \Box p)$ we can obtain $Red[G]$ as follows

$$\begin{aligned} Red[G] = {} & \mathbf{A}\Box(x_1 \equiv \mathbf{E}\Box p) \wedge \\ & \mathbf{A}\Box(x_2 \equiv \mathbf{E}\Diamond x_1) \wedge \\ & \mathbf{A}\Box(x_3 \equiv \mathbf{E}(\Box\Diamond\neg p \wedge \bigcirc q)) \wedge \\ & \mathbf{A}\Diamond(x_3 \vee x_2) \end{aligned}$$

Proposition 2 (Correctness of the procedure *Red*). For any ECTL$^+$ formula C, $\langle \mathcal{M}, s_0 \rangle \models C$ if, and only if, there exists a model $\mathcal{M}'$ such that $\langle \mathcal{M}', s_0 \rangle \models Red(C)$, where *Red* is introduced in Definition 2 [12].

Recall that the logic ECTL$^+$ extends ECTL similar to the way how CTL$^+$ extends CTL, namely, by allowing Boolean combination of temporal operators (but not any nesting of them). However, while CTL$^+$ is as expressive as CTL [10], ECTL$^+$ is strictly more expressive than ECTL. In our transformation procedure (see section 5) we will utilize equivalences used for CTL$^+ \longrightarrow$ CTL reduction, referring the reader to [10] for other cases which involve the $\mathcal{U}$ operation. (In the formulae below B and C are purely classical expressions.)

$$\begin{array}{l}
(a)\ \mathbf{P}(\bigcirc B \wedge \bigcirc C) \equiv \mathbf{P}\bigcirc(B \wedge C)\\
(b)\ \mathbf{P}(\bigcirc B \vee \bigcirc C) \equiv \mathbf{P}\bigcirc(B \vee C)\\
(c)\ \mathbf{P}(\square B \wedge \square C) \equiv \mathbf{P}\square(B \wedge C)\\
(d)\ \mathbf{P}(\diamondsuit B \vee \diamondsuit C) \equiv \mathbf{P}\diamondsuit(B \vee C)\\
(e)\ \mathbf{E}(B \vee C) \equiv \mathbf{E}B \vee \mathbf{E}C\\
(f)\ \mathbf{A}(B \wedge C) \equiv \mathbf{A}B \wedge \mathbf{A}C
\end{array} \tag{4}$$

Like ECTL, ECTL$^+$ allows limited nesting of temporal operators to express fairness constraints. For some of them, namely, for $\mathbf{A}\square\diamondsuit$ and $\mathbf{E}\diamondsuit\square$ cases, the validity of the following equivalences can be easily shown:

$$\begin{array}{l}
\text{(a)}\ \mathbf{A}\square\diamondsuit B \equiv \mathbf{A}\square\mathbf{A}\diamondsuit B\\
\text{(b)}\ \mathbf{E}\diamondsuit\square B \equiv \mathbf{E}\diamondsuit\mathbf{E}\square B
\end{array} \tag{5}$$

Next, we will give a number of obvious ECTL$^+$ equivalences that can be used in simplifying formulae.

$$\begin{array}{rlrl}
\mathbf{PT}(\mathbf{false}) \equiv & \mathbf{false} & \mathbf{PT}(\mathbf{true}) \equiv & \mathbf{true}\\
\mathbf{P}(R\,\mathcal{U}\,\mathbf{false}) \equiv & \mathbf{false} & \mathbf{P}(R\,\mathcal{U}\,\mathbf{true}) \equiv & \mathbf{true}\\
\mathbf{P}(\mathbf{false}\,\mathcal{U}\,R) \equiv & R & \mathbf{P}(\mathbf{true}\,\mathcal{U}\,R) \equiv & \mathbf{P}\diamondsuit R\\
\mathbf{P}(R\,\mathcal{W}\,\mathbf{false}) \equiv & \mathbf{P}\square R & \mathbf{P}(R\,\mathcal{W}\,\mathbf{true}) \equiv & \mathbf{true}\\
\mathbf{P}(\mathbf{false}\,\mathcal{W}\,R) \equiv & R & \mathbf{P}(\mathbf{true}\,\mathcal{W}\,R) \equiv & \mathbf{true}
\end{array} \tag{6}$$

where $\mathbf{P}$ is either of path quantifiers and $\mathbf{T}$ is either of the unary temporal operators.

We will use these equivalences in our transformation procedure, see section 5.

Further, applying procedure $\mathrm{NNF}_{\mathrm{ECTL}^+}$ and standard classical logic transformations, we can obtain for any ECTL$^+$ formula F (that has the degree of path quantifiers nesting 1) its 'special' Disjunctive or Conjunctive Normal Form, abbreviated as $DNF_{\mathbf{E}}(F)$ and $CNF_{\mathbf{A}}(F)$.

Definition 7. ($DNF_{\mathbf{E}}$ and $CNF_{\mathbf{A}}$ for ECTL$^+$ formulae). Let us call formulae of the type $\mathbf{T}(F_1)$, $F_1\mathbf{T}^2F_2$, $\diamondsuit\square F_1$, $\square\diamondsuit F_1$ (where F_1 and F_2 are purely classical) as elementary formulae. Now, a formula in $DNF_{\mathbf{E}}$ is of the type $\mathbf{E}(\alpha_1 \vee \ldots \vee \alpha_n)$ and a formula in $CNF_{\mathbf{A}}$ is of the type $\mathbf{A}(\alpha_1 \wedge \ldots \wedge \alpha_n)$, where each $\alpha_i (1 \leqslant i \leqslant n)$ is an elementary formula.

For example, the following formula (which we considered in section 2 – See Formula (3)) $\mathbf{A}(\square \diamondsuit p \wedge \diamondsuit \square \neg p)$ is in $CNF_{\mathbf{A}}$. The proof of the following proposition can be established immediately from the semantics of ECTL$^+$.

Proposition 3 (Correctness of the *DNF*$_{\mathbf{E}}$ and *CNF*$_{\mathbf{A}}$). For any ECTL$^+$ formula F which starts with $\mathbf{A}$ ($\mathbf{E}$) and has the degree of path quantifiers nesting 1, there exists its $DNF_{\mathbf{E}}(F)$ ($CNF_{\mathbf{A}}(F)$) such that F is satisfiable if and only if $DNF_{\mathbf{E}}(F)$ and $CNF_{\mathbf{A}}(F)$ are satisfiable, respectively.

Similar to ECTL, a class of *basic* ECTL$^+$ *modalities* consists of basic CTL modalities, enriched by the fairness constrains, $\mathbf{P}\square\diamondsuit$ and $\mathbf{P}\diamondsuit\square$. Our translation to SNF$_{\text{CTL}}$ and temporal resolution rules are essentially based upon the fixpoint characterizations of basic CTL modalities (see [7]) namely, upon definition of $\mathbf{P}\square$ and $\mathbf{P}\mathcal{W}$ as maximal fixpoints and $\mathbf{P}\diamondsuit$ and $\mathbf{P}\mathcal{U}$ as minimal fixpoint. Thus, $\mathbf{E}\square p$, $\mathbf{A}\square p$, $\mathbf{E}(p\mathcal{W}q)$, and $\mathbf{A}(p\mathcal{W}q)$ can be understood as maximal fixpoints represented by equations (7)–(10), respectively, while $\mathbf{E}\diamondsuit p$, $\mathbf{A}\diamondsuit p$ and $\mathbf{E}(p\,\mathcal{U}\,q)$, and $\mathbf{A}(p\,\mathcal{U}\,q)$ are given as minimal fixpoints represented by equations (11)–(14), respectively, (here, 'ν' is maximal fixpoint operator and 'μ' is minimal fixpoint operator).

$$\mathbf{E}\square p = \nu\zeta(p \wedge \mathbf{E}\bigcirc\zeta) \tag{7}$$

$$\mathbf{A}\square p = \nu\eta(p \wedge \mathbf{A}\bigcirc\eta) \tag{8}$$

$$\mathbf{E}(p\,\mathcal{W}\,q) = \nu\kappa(q \vee (p \wedge \mathbf{E}\bigcirc\kappa)) \tag{9}$$

$$\mathbf{A}(p\,\mathcal{W}\,q) = \nu\xi(q \vee (p \wedge \mathbf{A}\bigcirc\xi)) \tag{10}$$

$$\mathbf{E}\diamondsuit p = \mu\rho(p \vee \mathbf{E}\bigcirc\rho) \tag{11}$$

$$\mathbf{A}\diamondsuit p = \mu\tau(p \vee \mathbf{A}\bigcirc\tau) \tag{12}$$

$$\mathbf{E}(p\,\mathcal{U}\,q) = \mu\chi(q \vee (p \wedge \mathbf{E}\bigcirc\chi)) \tag{13}$$

$$\mathbf{A}(p\,\mathcal{U}\,q) = \mu\delta(q \vee (p \wedge \mathbf{A}\bigcirc\delta)) \tag{14}$$

Next we recall some results on interpreting CTL-type branching time logics over so-called *canonical models*. We will formulate these general results in relation to the logic ECTL$^+$, noting that they cover all CTL-type logics, including CTL*.

Since underlying models for ECTL$^+$ are countable ω trees, a state in such a model can have an infinite number of successor states. However, following [12] (Theorem 3.2), if a formula F is satisfiable in a CTL* (hence ECTL$^+$) model then it has a (finite) model, where each state has a branching degree $\leqslant |F|$ (where $\leqslant |F|$ is the length of F).

Definition 8 (Branching factor of a tree structure). Given the set $\mathcal{K} = \{k_1, k_2, \ldots k_n\}$, of the branching degrees of the states of a tree, the maximal $k_i (1 \leqslant i \leqslant n)$ is called the branching factor of this tree.

Definition 9 (Labelled tree). Given a tree $\mathcal{T} = (S, R)$ (where S is a set of nodes and R is a set of edges) and a finite alphabet, Σ, a Σ-labelled tree is a structure $(\mathcal{T}, \mathcal{L})$ where $\mathcal{L}$ is a mapping $S \longrightarrow \Sigma$, which assigns for each state, element of S, some label, element of Σ.

Observe that in section 2.2 we introduced the notion of satisfiability and validity of ECTL$^+$ formulae in relation to $\langle \mathcal{M}, s_0 \rangle$. Now, let us, following [16], call such a structure a *tree interpretation*.

Next we recall a notion of a k-ary tree canonical model which plays a fundamental role in our correctness argument. For these purposes, again following [16] and preserving its notation, we will look at tree interpretations as *tree generators*: the root of the tree is understood as an empty string, λ, and the whole tree is seen as a result of unwinding of the root applying the successor function $\{(s, si) | s \in [k]^\star, i \in \mathcal{K}\}$, where $[k]^\star = S$ and $si(i \in \mathcal{K})$ is a set of successors of a state s.

Definition 10 (Tree canonical interpretation). Let $\mathcal{T} = (S, R)$ be a k-ary infinite tree such that $[k]$ denotes the set $\{1, \ldots, k\}$, of branching degrees of the states in S and $R = \{(s, si) | s \in [k]^\star, i \in \mathcal{K}\}$. Now, given an alphabet $\Sigma = 2^{Prop}$, a k-ary tree canonical interpretation for an ECTL $^+$ formula F is of the form $\langle \mathcal{M}, \lambda \rangle$, where $\mathcal{M} = ([k]^\star, R, \pi)$ such that $\pi : [k]^\star \longrightarrow 2^{Prop}$ is a function which assigns truth values to the atomic propositions in each state.

As it is stated in [16], since in a canonical interpretation $\langle ([k]^\star, R, \pi), \lambda \rangle$, "the set of states, the initial state and the successor relation are all fixed they reduce to a function $[k]^\star \longrightarrow 2^{Prop}$, that is to a labelled tree over the alphabet 2^{Prop}". We will refer to this tree as a *canonical model*. Proposition 4 given below collects the results of [16] (Lemma 3.5, page 145).

Proposition 4 (Existence of a canonical model for ECTL$^+$). If an ECTL$^+$ formula F containing n (existential) path quantifiers has a model, then it has an $(n+1)$-*ary* canonical model.

Thus, given an interpretation $\langle \mathcal{M}, s_0 \rangle$ for an ECTL$^+$ formula F, there exists an $(n+1)$-ary canonical tree interpretation $\langle \mathcal{M}', \lambda \rangle$, where n is the number of existential path quantifiers in F, such that F is satisfied in $\langle \mathcal{M}, s_0 \rangle$ iff F is satisfied in $\langle \mathcal{M}', \lambda \rangle$.

These results were essentially used in the formulation of the transformation rule for the ECTL fairness constraint $\mathbf{A} \diamondsuit \square$ [3, 4]. In this paper we will further extend their applicability in the transformation procedure for ECTL$^+$.

4. Normal form for ECTL$^+$

As a normal form for ECTL$^+$, similarly to ECTL, we utilise a clausal normal form, defined for the logic CTL, SNF$_{\mathrm{CTL}}$, which was developed in [2, 6]. The core idea of SNF$_{\mathrm{CTL}}$ is to extract from a given formula the following three types of constraints. *Initial constraints* represent information relevant to the initial moment of time, the root of a tree. *Step constraints* indicate what will happen at the successor state(s) given that some conditions are satisfied 'now'. Finally, *Sometime constraints* keep track on any eventuality, again, given that some conditions are satisfied 'now'. Therefore, similar to the linear-time case ([14]) an important part of the transformation procedure for ECTL$^+$ formulae into SNF$_{\mathrm{CTL}}$ is the removal of all other, 'unwanted' modalities $\mathbf{A}\square, \mathbf{E}\square, \mathbf{A}\mathcal{U}, \mathbf{E}\mathcal{U}$, $\mathbf{A}\mathcal{W}$, $\mathbf{E}\mathcal{W}$ (see section 5.2).

Additionally, to preserve a specific path context during the translation, we incorporate indices.

Indices. The language for indices is based on the set of terms

$$\mathsf{IND} = \{\langle \mathsf{f}\rangle, \langle \mathsf{g}\rangle, \langle \mathsf{h}\rangle, \langle LC(\mathsf{f})\rangle, \langle LC(\mathsf{g})\rangle, \langle LC(\mathsf{h})\rangle \ldots\}$$

where $\mathsf{f}, \mathsf{g}, \mathsf{h} \ldots$ denote constants. Thus, $\mathbf{E}A_{\langle \mathsf{f}\rangle}$ means that A holds on some path labelled as $\langle \mathsf{f}\rangle$. A designated type of indices in SNF$_{\mathrm{CTL}}$ are indices of the type $\langle \mathsf{LC(ind)}\rangle$ which represent a limit closure of prefixes associated with $\langle \mathsf{ind}\rangle$. All formulae of SNF$_{\mathrm{CTL}}$ of the type $P \Rightarrow \mathbf{E}\bigcirc Q$ or $P \Rightarrow \mathbf{E}\Diamond Q$, where Q is a purely classical expression, are labelled with some index. Labelling clauses of the normal form by indices makes paths explicit and is related to the branching factor of the canonical model for the clauses and will be explained later.

The SNF$_{\mathrm{CTL}}$ language is obtained from the ECTL$^+$ language by omitting the $\mathcal{U}$ and $\mathcal{W}$ operators, and adding classically defined constants **true** and **false**, and a new operator, **start** ('at the initial moment of time') defined as

$$\langle \mathcal{M}, s_i\rangle \models \mathbf{start} \quad \text{iff} \quad i = 0$$

Definition 11 (Separated Normal Form SNF$_{\mathbf{CTL}}$). SNF$_{\mathrm{CTL}}$ is a set of formulae $\mathbf{A}\square\left[\bigwedge_i (P_i \Rightarrow F_i)\right]$ where each of the clauses $P_i \Rightarrow F_i$ is further restricted as below, each $\alpha_j, \alpha_p, \alpha_t, \alpha_v$, $\beta_i, \beta_m, \beta_r$ or γ is a literal, **true** or **false** and $\langle \mathsf{ind}\rangle \in \mathsf{IND}$ is some index.

$\mathbf{start} \Rightarrow \bigvee_{i=1}^{k} \beta_i$	an initial clause
$\bigwedge_{j=1}^{l} \alpha_j \Rightarrow \mathbf{A}\bigcirc\left[\bigvee_{m=1}^{n} \beta_m\right]$	an **A** step clause
$\bigwedge_{p=1}^{q} \alpha_p \Rightarrow \mathbf{E}\bigcirc\left[\bigvee_{r=1}^{s} \beta_r\right]_{\langle \mathsf{ind}\rangle}$	an **E** step clause
$\bigwedge_{t=1}^{u} \alpha_t \Rightarrow \mathbf{A}\Diamond\gamma$	an **A** sometime clause
$\bigwedge_{v=1}^{w} \alpha_v \Rightarrow \mathbf{E}\Diamond\gamma_{\langle \mathsf{LC(ind)}\rangle}$	an **E** sometime clause

Interpreting $\mathbf{SNF_{CTL}}$. Let $\mathrm{SNF_{CTL}}(G)$ be a set of $\mathrm{SNF_{CTL}}$ clauses obtained for some ECTL$^+$ formula G with n existential path quantifiers. As we will see (section 5), at some stage of the transformation of G into $\mathrm{SNF_{CTL}}(G)$ we associate every $k \in 1 \ldots n$ with a unique index $ind_k \in \mathsf{IND}$ and label each $\mathbf{E}$ step clause with the unique ind_k and each $\mathbf{E}$ sometime clause with the unique $LC(ind_k)$ (which can be justified by Proposition 4).

The underlying models of $\mathrm{SNF_{CTL}}$ similar to ECTL$^+$ are countable ω trees and we obtain the $\mathrm{SNF_{CTL}}$ semantics from the semantics of ECTL$^+$ (section 2) by preserving items s1–s7, and p3–p4. The natural intuition here is that the initial clauses provide starting conditions while step and sometime clauses constrain the future behaviour. An initial $\mathrm{SNF_{CTL}}$ clause, $\mathbf{start} \Rightarrow F$, is understood as " F is satisfied at the initial state of some model $\mathcal{M}$." Any other $\mathrm{SNF_{CTL}}$ clause is interpreted taking also into account that it occurs in the scope of $\mathbf{A}\square$.

Thus, a clause $\mathbf{A}\square(x \Rightarrow \mathbf{A}\bigcirc p)$ (see figure 3) is interpreted as "for any fullpath χ and any state $s_i \in \chi(0 \leqslant i)$, if x is satisfied at a state s_i then p must be satisfied at the moment, next to s_i, along each path which starts from s_i".

Next, a clause $\mathbf{A}\square(x \Rightarrow \mathbf{E}\bigcirc q_{\langle \mathsf{ind} \rangle})$ (a model for which is given again in figure 3) is interpreted as "for any fullpath χ and any state $s_i \in \chi(0 \leqslant i)$, if x is satisfied at a state s_i then q must be satisfied at the moment, next to s_i, along a path which starts from s_i and which is associated with ind". Speaking informally, we interpret $\mathbf{A}\square(x \Rightarrow \mathbf{E}\bigcirc q_{\langle \mathsf{ind} \rangle})$ such that given a state in a model which satisfies x (the left hand side of the clause), the label, ind, indicates the direction, in which the successor state which satisfies q can be reached (see similar developments in the construction of logic DCTL*, [15]).

Finally, the labelling of the $\mathbf{E}$ sometime clause is justified based upon its fixpoint characterization. Consider $\mathbf{A}\square(x \Rightarrow \mathbf{E}\Diamond p_{\langle \mathsf{LC(ind)} \rangle})$. This has the following meaning "for any fullpath χ and any state $s_i \in \chi(0 \leqslant i)$, if x is satisfied at a state s_i then p must be satisfied at some state, say $s_j (i \leqslant j)$, along some path α_{s_i} which is the limit closure of $\langle \mathsf{ind} \rangle$ which departs from s_i". Note that our interpretation of an LC index corresponds to the concept of a linear interpretation [16]. Speaking informally, the meaning of the index $\mathsf{LC(ind)}$ is to indicate that once x is satisfied at some state, say s_i, of a model, a state which satisfies p, say s_j, becomes reachable from s_i along some path

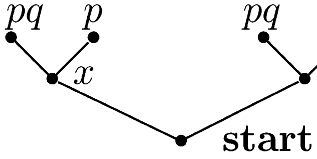

Figure 3. Interpretation of step and sometime clauses.

associated with $\mathsf{LC}(\mathsf{ind})$. This corresponds to the following understanding of $\mathbf{E}\diamondsuit$ formulae based upon the equivalence (11)

$$\mathbf{E}\diamondsuit p \equiv p \vee \mathbf{E}\bigcirc\mathbf{E}\diamondsuit p \tag{15}$$

Thus, given $\mathbf{E}\diamondsuit p$ and recursively applying (15) we derive that either p is satisfied at s_i or, alternatively, there is a successor of s_i, say s_j, which satisfies $\mathbf{E}\diamondsuit p$. Now, again, either s_j satisfies p or it has a successor, say s_k, which satisfies $\mathbf{E}\diamondsuit p$, etc. In the canonical model we are additionally assured that s_j is an ind-th successor of s_i, s_k is an ind-th successor of s_j, etc, and thus the state satisfying p at which we eventually arrive can be reached from s_i on a linear path which is a limit closure of ind.

5. Transformation of ECTL$^+$ formulae into SNF$_{\mathrm{CTL}}$

In this section we first describe the algorithm to transform ECTL$^+$ formulae into SNF$_{\mathrm{CTL}}$, then we present rules involved into various stages of the algorithm, and, finally, give an example transformation.

5.1. *Algorithm to transform ECTL$^+$ formulae into SNF$_{CTL}$*

As SNF$_{\mathrm{CTL}}$ is a part of the resolution technique, to check validity of an ECTL$^+$ formula G, we first negate the latter, translate $\neg G$ into its Negation Normal Form deriving $\mathrm{NNF}_{\mathrm{ECTL}}(\neg G)$ and simplify the latter. Let C be the result of these transformations. We introduce the transformation procedure $\tau = [\tau_2[\tau_1[C]]]$ applied to C, where τ_1 and τ_2 are described, respectively, by the steps 1–2 and 3–10 below.

(1) Anchor C to **start** and rename C by a new proposition, say, x_0 obtaining

$$\mathbf{A}\,\square\,(\mathbf{start} \Rightarrow x_0) \wedge \mathbf{A}\square(x_0 \Rightarrow C)$$

(2) Apply equivalences (5) and procedure *Red* (see Definition 2) to C. Thus, we derive a set of constraints of the following structure

$$\mathbf{A}\,\square\left[(\mathbf{start} \Rightarrow x_0) \wedge \left[\bigwedge_{j=0}^{m}(P_j \Rightarrow Q_j)\right]\right]$$

where P_j is a proposition, Q_j is either a purely classical formula or if Q_j contains a path quantifier then the degree of nesting of path quantifiers in Q_j is 1.

Let us call a formula G in *pre-clause form* if $\tau_1[G] = G$ i.e., it is of the form $P_j \Rightarrow Q_j$ where P_j is a literal, conjunction of literals, or **start**, Q_j is a purely classical formula or any of $\mathbf{PT}C_j, \mathbf{P}\Diamond\Box C_j, \mathbf{P}\Box\Diamond C_j, \mathbf{P}(C_{j_1}\mathbf{T}^2C_{j_2}), \mathbf{P}(\mathbf{T}C_{j_1} \wedge \ldots \wedge \mathbf{T}C_{j_n}), \mathbf{P}(\mathbf{T}C_{j_1} \vee \ldots \vee \mathbf{T}C_{j_m})$ (for some $n, m \geqslant 1$) and $C_j, C_{j_1}, \ldots$ are purely classical formulae.

(3) For every pre-clause $P_j \Rightarrow Q_j$:
  (3.1) If Q_j is an ECTL$^+$ formula but not a CTL$^+$ formula then do the following:
    (3.1.1) obtain its $DNF_{\mathbf{E}}(Q_j)$ or $CNF_{\mathbf{A}}(Q_j)$ and apply equivalences 4-(e) or 4-(f), respectively.
    (3.1.2) apply equivalences (5).
    (3.1.3) apply procedure *RED*.
  (3.2) If Q_j contains Boolean combinations of temporal operators but does not contain any fairness constraint then (as it is a CTL$^+$ formula) apply the procedure to transform CTL$^+$ into CTL (see section 3).

(4) At this stage, renaming state subformulae (which are expressed by basic CTL modalities) on the right hand-sides of the constraints derived at step 3 we obtain the structure required for a pre-clause.

(5) For every pre-clause $P_j \Rightarrow Q_j$, by continuous renaming of the embedded classical subformulae by auxiliary propositions together with some classical transformations we obtain the following conditions.

– If Q_j contains a basic CTL modality then

– If $Q_j = \mathbf{PT}C_j$ and $\mathbf{PT}$ is not $\mathbf{P}\bigcirc$ then C_j is a literal, else C_j is a purely classical formula.

– If $Q_j = \mathbf{E}\Box\Diamond C_j$ or $Q_j = \mathbf{A}\Diamond\Box C_j$ then C_j is a literal,

– If $Q_j = \mathbf{P}(C_{j_1}\mathbf{T}^2C_{j_2})$ then C_{j_1} and C_{j_2} are literals.

– If $Q_j = \mathbf{E}(\alpha_1 \wedge \ldots \wedge \alpha_n)$ or $Q_j = \mathbf{A}(\beta_1 \vee \ldots \vee \beta_m)$, where each $\alpha_k (1 \leqslant k \leqslant n)$ and $\beta_l (1 \leqslant l \leqslant m)$ is a temporal operator or a fairness constraint applied to classical formulae (but not literals) we obtain the structure where they apply to literals.

(6) Label each pre-clause containing the $\mathbf{E}\bigcirc$ modality by an unique index $\langle \mathsf{ind_i} \rangle \in \mathsf{IND}$ and any other pre-clause containing the $\mathbf{E}$ quantifier by an unique index $\langle \mathsf{LC(ind_j)} \rangle \in \mathsf{IND}$. Let *LIST_IND* be a list of all indices introduced during this labelling.

(7) Transform pre-clauses with $\mathbf{E}\Box\Diamond$ and $\mathbf{A}\Diamond\Box$.

(8) Transform pre-clauses containing $\mathbf{E}(\alpha_1 \wedge \ldots \wedge \alpha_n)$ or $\mathbf{A}(\beta_1 \vee \ldots \vee \beta_m)$ (of the structure obtained at step (5)).

(9) Remove all unwanted basic CTL modalities.

(10) Derive the desired form of SNF$_{\text{CTL}}$ clauses. At this final stage we transform pre-clauses $P_j \Rightarrow Q_j$, where Q_j is either $\mathbf{P}\bigcirc C_j$ or a purely classical formula: for every pre-clause $P_j \Rightarrow \mathbf{P}\bigcirc C_j$, we obtain the structure where $\mathbf{P}\bigcirc$ applies either to a literal or to disjunction of literals. This can be achieved, again, by renaming of the embedded classical subformulae, translating C'_j into conjunctive normal form (CNF), and distributing $\mathbf{P}\bigcirc$ over conjunction, together with some classical transformations. Further, for every remaining purely classical pre-clause $P_j \Rightarrow Q_j$, we apply a number of procedures including those that are used in classical logic in transforming formulae to CNF, some simplifications and the introduction of a temporal context.

5.2. *Transformation rules towards SNF$_{CTL}$*

In the presentation below we omit the outer ‘ $\mathbf{A}\square$’ connective that surrounds the conjunction of formulae and, for convenience, consider a set of formulae rather than the conjunction.

Simplification rules. Several classical simplification rules are used at different stages of the transformation procedure.

To simplify formulae with **false** or **true** constrained as arguments of a basic ECTL$^+$ modality we use the following equivalences (below R is any ECTL$^+$ formula).

$$\begin{array}{ll} (R \wedge \mathbf{true}) \equiv R & (R \wedge \mathbf{false}) \equiv \mathbf{false} \\ (R \vee \mathbf{true}) \equiv \mathbf{true} & (R \vee \mathbf{false}) \equiv R \\ (R \Rightarrow \mathbf{true}) \equiv \mathbf{true} & (\mathbf{false} \Rightarrow R) \equiv \mathbf{false} \end{array} \tag{16}$$

Also, in simplifying formulae we utilize the equivalences (6) and standard rules applied to Boolean connectives used to obtain CNF and DNF.

Next, for any ECTL$^+$ formulae Q and S, we split conjuncts:

$$\frac{P \Rightarrow (Q \wedge S)}{\begin{array}{l} P \Rightarrow Q \\ P \Rightarrow S \end{array}} \tag{17}$$

Further, if Q is embedded within $\mathbf{P}\bigcirc Q$ and is not of the form $Q_1 \wedge Q_2$, then, since Q is classical, we translate Q into CNF, applying the standard set of rules required for such transformations.

Alternatively, i.e., if Q is of the form $Q_1 \wedge Q_2$, then we distribute $\mathbf{P}\bigcirc$ over conjunction. Here we distinguish two cases, when $\bigcirc$ is preceded by an $\mathbf{A}$ quantifier and by a $\mathbf{E}$ quantifier. In the first case the rule applies without any restrictions as follows.

$$\frac{P \Rightarrow \mathbf{A}\bigcirc(Q_1 \wedge Q_2)}{\begin{array}{l} P \Rightarrow \mathbf{A}\bigcirc Q_1 \\ P \Rightarrow \mathbf{A}\bigcirc Q_2 \end{array}} \tag{18}$$

However, in the second case, when $\bigcirc$ is paired with the existential quantifier, we must follow our requirement on preserving labelling of $\mathbf{E}$-formulae.

$$\frac{P \Rightarrow \mathbf{E}\bigcirc(Q_1 \wedge Q_2)_{\langle \mathsf{ind} \rangle}}{\begin{array}{l} P \Rightarrow \mathbf{E}\bigcirc Q_{1\langle \mathsf{ind} \rangle} \\ P \Rightarrow \mathbf{E}\bigcirc Q_{2\langle \mathsf{ind} \rangle} \end{array}} \tag{19}$$

The fact that both conclusions of this rule are labelled by the same index will be useful in enabling the application of step resolution rules and also in searching for loops as a part of the temporal resolution method (section 8).

Next, we present a transformation rule for purely classical formulae.

Temporizing. Given a purely classical formula $P \Rightarrow Q$, we introduce a temporal context applying a rule called *Temporizing*:

$$\frac{P \Rightarrow Q}{\begin{array}{r} \mathbf{start} \Rightarrow \neg P \vee Q \\ \mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee Q) \end{array}} \tag{20}$$

A particular case of the temporising rule applies to a purely classical expression with **false** as its consequent, for example, $P \Rightarrow \mathbf{false}$. Here temporizing gives us a set of formulae $\{\mathbf{start} \Rightarrow \neg P \vee \mathbf{false}, \mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee \mathbf{false})\}$ which can be further simplified to $\{\mathbf{start} \Rightarrow \neg P, \mathbf{true} \Rightarrow \mathbf{A}\bigcirc\neg P\}$.

Renaming rule. Renaming applies at various stages of the transformation procedure. Since renaming involves replacing a subformula R within some complex formula F by a new proposition symbol, x, it must be accompanied by associating the truth values of x with the truth values of R in every model $\mathcal{M}$. In branching-time framework, this link between the truth values of x and R must be provided in every state along every path of the underlying model structure, i.e., $\models \mathbf{A}\Box(x \equiv R)$ must hold. Note that in the standard case, in general, a complex formula R to be renamed can be embedded into F as its subformula under either positive or negative polarity. Therefore, in providing the required link between the renamed subformula and a new variable used for renaming, we must utilize the '$\equiv$' operator. Below we give the inductive definition of polarity of the embedded ECTL$^+$ formulae adapting here the standard definition.

Definition 12 (Polarity).

- Let p be an atomic proposition. Then p occurs *positively* (*negatively*) in p ($\neg p$).
- If $A \wedge B$ or $A \vee B$ positively (negatively) occur in C then both A and B positively (negatively) occur in C.
- If $\neg A$ positively (negatively) occurs in C then A negatively (positively) occurs in C.
- If $A \Rightarrow B$ positively (negatively) occurs in C then A negatively (positively) and B positively (negatively) occurs in C.
- Let $\mathbf{P}$ abbreviate either of the path quantifiers, $\mathbf{T}$ abbreviate any unary and $\mathbf{T}^2$ either of binary temporal operators, and A and B be state formulae.

  Now,

  - if $\mathbf{PT}A$ or $\mathbf{P}(A\mathbf{T}^2B)$ positively (negatively) occurs in C then both A and B positively (negatively) occur in C.
  - if $\mathbf{P}\Box\Diamond A$ or $\mathbf{P}\Diamond\Box A$ positively (negatively) occurs in C then A positively (negatively) occur in C.

Observe, however, that, according to the transformation algorithm, a complex $\mathbf{P}$-embedded subformula B, to which we apply the renaming operation, occurs on the right hand side of some formula $R_i = (A \Rightarrow B)$. Further, since the procedure NNF$_{\mathrm{CTL}}$ has been already applied, this occurrence of B within R_i is always positive. Therefore, to prevent the enlargement of the complexity of τ, we can accompany the renaming by establishing the link between B and a new variable x used in renaming of B, by requiring $\models \mathbf{A}\Box(x \Rightarrow B)$, rather than $\models \mathbf{A}\Box(x \equiv B)$, thus, not duplicating renamed subformulae.

Now, the general rule for renaming of the embedded ECTL$^+$ state subformulae is as follows:

$$\frac{P \Rightarrow \mathcal{P}(R)}{\begin{array}{l} P \Rightarrow \mathcal{P}(R/x) \\ x \Rightarrow R \end{array}}$$

where P is either a purely classical expression or **start** and $\mathcal{P}(R)$ is an ECTL$^+$ formula with the designated state subformula R and $\mathcal{P}(R/x)$ means a result of replacing R by a new propositional symbol x in $\mathcal{P}$.

Recall that after obtaining $\mathrm{NNF}_{\mathrm{CTL}}(\neg G)$ for some input G, we simplify the latter and anchor the result, C, to **start** deriving **start** $\Rightarrow C$. Next we rename C by some new proposition (initial renaming). In other cases we apply the renaming technique to reduce the nesting of P-embedded subformuale (as part of the procedure *Red*), to obtain arguments of basic ECTL$^+$ modalities as literals (by renaming complex classical subformulae embedded within the scope of basic CTL modalities excluding $\mathbf{P}\bigcirc$), and, finally, we utilize renaming to manage embedded path subformulae.

Managing embedded path subformulae in ECTL$^+$. We incorporate rules to rename purely path formulae embedded in ECTL$^+$ fairness constraints from [3]. Let the number of indices in *LIST_IND* be $n (n \geqslant 0)$ and let $\langle \mathsf{ind_1} \rangle, \ldots \langle \mathsf{ind_n} \rangle \in IND$ be the constants occurring in these indices. If, however, for some index $\langle \mathsf{ind} \rangle \in$ *LIST_IND* we do not have $\langle LC(\mathsf{ind}) \rangle \in$ *LIST_IND* then we upgrade *LIST_IND* by $\langle LC(\mathsf{ind}) \rangle$ (in the formulation below n is the number of indices in *LIST_IND* and $x, x_1, \ldots, x_n$ are new propositions).

Renaming: E $\square$ $\diamondsuit$ case.

$$\frac{P \Rightarrow \mathbf{E} \square \diamondsuit Q_{\langle LC(\mathsf{ind}) \rangle}}{\begin{array}{l} P \Rightarrow \mathbf{E} \square\, x_{\langle LC(\mathsf{ind}) \rangle} \\ x \Rightarrow \mathbf{E} \diamondsuit Q_{\langle LC(\mathsf{ind}) \rangle} \end{array}}$$

Renaming: The **A $\diamondsuit$ $\square$ case**.

if $n = 0$

$$\frac{P \Rightarrow \mathbf{A} \diamondsuit \square\, Q}{\begin{array}{l} P \Rightarrow \mathbf{E} \diamondsuit\, x_{\langle LC(\mathsf{ind}) \rangle} \\ x \Rightarrow \mathbf{E} \square\, Q_{\langle LC(\mathsf{ind}) \rangle} \end{array}}$$

if $n > 0$

$$\frac{P \Rightarrow \mathbf{A} \diamondsuit \square Q}{\begin{array}{l} P \Rightarrow \mathbf{E} \diamondsuit\, x_{1 \langle LC(\mathsf{ind_1}) \rangle} \\ x_1 \Rightarrow \mathbf{E} \square\, Q_{\langle LC(\mathsf{ind_1}) \rangle} \\ \ldots \\ P \Rightarrow \mathbf{E} \diamondsuit\, x_{n \langle LC(\mathsf{ind_n}) \rangle} \\ x_n \Rightarrow \mathbf{E} \square\, Q_{\langle LC(\mathsf{ind_n}) \rangle} \end{array}}$$

Indices. Recall that at step 6 of the transformation procedure, we introduce labelling of the $\mathrm{SNF}_{\mathrm{CTL}}$ pre-clauses containing the $\mathbf{E}$ quantifier. The justification of this labelling is based upon fixpoint characterization of basic CTL modalities and was explained in [2, 3] except for the new specific ECTL$^+$ formulae in $DNF_{\mathbf{E}}$ form. The latter can be explained simply based upon the $\mathrm{SNF}_{\mathrm{CTL}}$ semantics.

Rules to remove basic CTL modalities. Removal rules are derived from the fixpoint characterization of basic CTL modalities (7)–(14) (see also [6]). In the formulation of these rules given below x is a new proposition:

Removal of $\mathbf{A}\,\square$

$$\frac{P \Rightarrow \mathbf{A}\,\square\, y}{\begin{array}{l} P \Rightarrow y \wedge x \\ x \Rightarrow \mathbf{A}\bigcirc(y \wedge x) \end{array}}$$

Removal of $\mathbf{E}\,\square$

$$\frac{P \Rightarrow \mathbf{E}\,\square\, y_{\langle LC(\mathsf{ind})\rangle}}{\begin{array}{l} P \Rightarrow y \wedge x \\ x \Rightarrow \mathbf{E}\bigcirc(y \wedge x)_{\langle \mathsf{ind}\rangle} \end{array}}$$

Removal of $\mathbf{A}\mathcal{U}$

$$\frac{P \Rightarrow \mathbf{A}(p\,\mathcal{U}\,q)}{\begin{array}{l} P \Rightarrow q \vee (p \wedge x) \\ x \Rightarrow \mathbf{A}\bigcirc(q \vee (p \wedge x)) \\ P \Rightarrow \mathbf{A}\,\diamondsuit\, q \end{array}}$$

Removal of $\mathbf{E}\,\mathcal{U}$

$$\frac{P \Rightarrow \mathbf{E}(p\,\mathcal{U}\,q)_{\langle LC(\mathsf{ind})\rangle}}{\begin{array}{l} P \Rightarrow q \vee (p \wedge x) \\ x \Rightarrow \mathbf{E}\bigcirc(q \vee (p \wedge x))_{\langle \mathsf{ind}\rangle} \\ P \Rightarrow \mathbf{E}\,\diamondsuit\, q_{\langle LC(\mathsf{ind})\rangle} \end{array}}$$

Removal of $\mathbf{A}\,\mathcal{W}$

$$\frac{P \Rightarrow \mathbf{A}(p\,\mathcal{W}\,q)}{\begin{array}{l} P \Rightarrow q \vee (p \wedge x) \\ x \Rightarrow \mathbf{A}\bigcirc(q \vee (p \wedge x)) \end{array}}$$

Removal of $\mathbf{E}\,\mathcal{W}$

$$\frac{P \Rightarrow \mathbf{E}(p\,\mathcal{W}\,q)_{\langle LC(\mathsf{ind})\rangle}}{\begin{array}{l} P \Rightarrow q \vee (p \wedge x) \\ x \Rightarrow \mathbf{E}\bigcirc(q \vee (p \wedge x))_{\langle \mathsf{ind}\rangle} \end{array}}$$

Managing embedded Boolean combinations of path subformulae in ECTL$^+$. Recall that on step 8 of the transformation procedure we must further reduce formulae of the form $\mathbf{E}(\alpha_1 \wedge \ldots \wedge \alpha_n)$ and $\mathbf{A}(\beta_1 \vee \ldots \vee \beta_m)$. The corresponding rules are given below where ind' is $\mathsf{LC}(\mathsf{ind})$ if the u_i are not $\bigcirc$, and ind otherwise, and n is the number of indices in *LIST_IND*.

$\mathbf{E}(\alpha_1 \wedge \ldots \wedge \alpha_n)_{\mathsf{ind}}$ **case**.

$$\frac{P \Rightarrow \mathbf{E}(\alpha_1 \wedge \ldots \wedge \alpha_n)_{\mathsf{ind}}}{\begin{array}{l} u_1 \Rightarrow \mathbf{E}(\alpha_1)_{\mathsf{ind}'} \\ \ldots \\ u_n \Rightarrow \mathbf{E}(\alpha_n)_{\mathsf{ind}'} \end{array}}$$

$\mathbf{A}(\alpha_1 \vee \ldots \vee \alpha_n)$**case**.

if $n = 0$

$$\frac{P \Rightarrow \mathbf{A}(\beta_1 \vee \ldots \vee \beta_m)}{P \Rightarrow \mathbf{E}\,(\beta_1 \vee \ldots \vee \beta_m)}$$

if $n > 0$

$$\frac{P \Rightarrow \mathbf{A}\,(\beta_1 \vee \ldots \vee \beta_m)}{\begin{array}{l} P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{ind_1} \\ \ldots \\ P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{ind_n} \end{array}}$$

5.3. *Example transformation*

As an example we translate into $\mathrm{SNF}_{\mathrm{CTL}}$ the formula:

$$\mathbf{E}(\Diamond \Box \neg p \vee \Box \Diamond p) \tag{21}$$

To check if (21) is valid we apply procedure $\mathrm{NNF}_{\mathrm{ECTL}^+}(\neg(\mathbf{E}(\Diamond \Box \neg p \vee \Box \Diamond p))) = \mathbf{A}(\Box \Diamond p \wedge \Diamond \Box \neg p)$ which was considered as an example of an unsatisfiable formula in section 2. From the translation algorithm, we derive steps 0–2, where x is a new proposition.

0. $\mathbf{start} \Rightarrow \mathbf{A}(\Box \Diamond p \wedge \Diamond \Box \neg p)$	anchoring to **start**
1. $\mathbf{start} \Rightarrow x$	0, Initial Renaming
2. $x \Rightarrow \mathbf{A}(\Box \Diamond p \wedge \Diamond \Box \neg p)$	0, Initial Renaming

We proceed with formula 2, where the right hand side of the implication is already in $CNF_{\mathbf{A}}(F)$. Thus, we apply equation (4)-(f) to distribute the $\mathbf{A}$ over conjunction in 2, obtaining 3, and then simplify the latter deriving 4 and 5. Next, we simplify formula 4 applying equation (5)-(a) to get 6. The structure of the latter enables us to apply procedure *Red* deducing 7 and 8 and introducing a new variable y.

3. $x \Rightarrow \mathbf{A}\Box \Diamond p \wedge \mathbf{A}\Diamond \Box \neg p$	2, equiv$(4 - f)$
4. $x \Rightarrow \mathbf{A}\Box \Diamond p$	3, SIMP
5. $x \Rightarrow \mathbf{A}\Diamond \Box \neg p$	3, SIMP
6. $x \Rightarrow \mathbf{A}\Box \mathbf{A}\Diamond p$	4, equiv$(5 - \mathrm{a})$
7. $x \Rightarrow \mathbf{A}\Box y$	6, Procedure Red
8. $y \Rightarrow \mathbf{A}\Diamond p$	6, Procedure Red

Applying the renaming rule ($\mathbf{A}\Diamond \Box$ case) to 5 we derive formula 9 and label it with a new index $\langle LC(\mathsf{f})\rangle$ (since *LIST_IND* is empty). Applying equation (5) to 9 we get $x \Rightarrow \mathbf{E}\Diamond \mathbf{E} \Box \neg p\langle LC(\mathsf{f})\rangle$ which is further reduced by procedure *Red* to 10 and 11, where z is a new variable. Apply $\mathbf{A}\,\Box$ removal rule to 7 and $\mathbf{E}\,\Box$ removal rule to 11, where x_1 and z_1 are new variables.

9. $x \Rightarrow \mathbf{E}\Diamond \Box \neg p_{\langle LC(\mathsf{f})\rangle}$	5, Renaming
10. $x \Rightarrow \mathbf{E}\Diamond z_{\langle LC(\mathsf{f})\rangle}$	9, Red
11. $z \Rightarrow \mathbf{E}\Box \neg p_{\langle LC(\mathsf{f})\rangle}$	9, Red
12. $x \Rightarrow y \wedge x_1$	7, Removal of $\mathbf{A}\,\Box$
13. $x_1 \Rightarrow \mathbf{A}\bigcirc(y \wedge x_1)$	7, Removal of $\mathbf{A}\,\Box$
14. $z \Rightarrow \neg p \wedge z_1$	11, Removal of $\mathbf{E}\,\Box$
15. $z_1 \Rightarrow \mathbf{E}\bigcirc(\neg p \wedge z_1)_{\langle \mathsf{f})\rangle}$	11, Removal of $\mathbf{E}\,\Box$

Next simplifying and temporising formulae 12 and 14 we obtain 16–19 and 20–23, respectively. Finally, we distribute $\mathbf{A}$ and $\mathbf{E}$ over $\bigcirc$ in 13 and 15.

16. $\mathbf{start} \Rightarrow \neg x \vee y$	12, Simp, Temp
17. $\mathbf{start} \Rightarrow \neg x \vee x_1$	12, Simp, Temp
18. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg x \vee y)$	12, Simp, Temp
19. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg x \vee x_1)$	12, Simp, Temp
20. $\mathbf{start} \Rightarrow \neg z \vee \neg p$	14, Simp, Temp
21. $\mathbf{start} \Rightarrow \neg z \vee z_1$	14, Simp, Temp
22. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg z \vee \neg p)$	14, Simp, Temp
23. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg z \vee \neg z_1)$	14, Simp, Temp
24. $x \Rightarrow \mathbf{A}\bigcirc y$	13, equiv $(4) - (\mathsf{a})$
25. $x \Rightarrow \mathbf{A}\bigcirc x_1$	13, equiv $(4) - (\mathsf{a})$
26. $z_1 \Rightarrow \mathbf{E}\bigcirc \neg p_{\langle \mathsf{f} \rangle}$	15, Distribution $\mathbf{E}\bigcirc$ over $\wedge$
27. $z_1 \Rightarrow \mathbf{E}\bigcirc z_{1\langle \mathsf{f} \rangle}$	15, Distribution $\mathbf{E}\bigcirc$ over $\wedge$

The normal form of the given ECTL$^+$ formula is represented by clauses 1, 8, 10, 16–27.

6. Correctness of the transformation of ECTL$^+$ formulae into SNF$_{\mathbf{CTL}}$

Here we provide the correctness argument for our transformation procedure. A significant part of this argument is either similar to the corresponding proofs given in [2, 3] for CTL and ECTL or extends these proofs for new cases of ECTL$^+$ formulae. Therefore, we will only state such claims referring the reader to [2, 3] while we sketch here proofs for new techniques used for ECTL$^+$ transformations. Note also that in our previous paper [3] we have not established the proof for the claim analogous to Lemma 3 (see below). Therefore, providing our argument in this paper, we not only show the desired correctness of the transformation procedure for ECTL$^+$ but also bridge this gap for ECTL.

Theorem 1. An ECTL$^+$ formula, G, is satisfiable if, and only if, $\tau(G)$ is satisfiable.

To establish the correctness of this theorem we first show that an ECTL$^+$ formula G is satisfiable, if and only if $\tau_1(G)$ is satisfiable (Lemma 1). At the next stage we prove that the transformation procedure τ_2 preserves satisfiability (Lemma 2). Finally, (Lemma 3), we show that given an ECTL$^+$ formula G and its normal form, SNF$_{\mathrm{CTL}}(G)$, if SNF$_{\mathrm{CTL}}(G)$ is satisfiable then G is satisfiable.

Lemma 1. An ECTL$^+$ formula, G, is satisfiable if, and only if, $\tau_1(G)$ is satisfiable.

Since τ_1 is taken from the translation of ECTL formulae to SNF$_{\mathrm{CTL}}$, the proof of Lemma 1 follows from the correctness argument for ECTL ([3]).

Lemma 2. Given an SNF$_{\mathrm{CTL}}$ formula G, if $\tau_1(G)$ is satisfiable then so is $\tau_2(\tau_1(G))$.

Here we must show that the new techniques used in our transformation procedure preserve satisfiability. This includes the correctness argument for $DNF_{\mathbf{E}}$ and $CNF_{\mathbf{A}}$ and also for the cases of Boolean combinations of temporal operators, $\mathbf{E}(\alpha_1 \wedge \ldots \wedge \alpha_n)_{\mathsf{ind}}$ case and $\mathbf{A}(\alpha_1 \vee \ldots \vee \alpha_n)$ case. Corresponding proofs are established straightforwardly from the SNF$_{\mathrm{CTL}}$ semantics, taking into account the meaning of indices and Proposition 4 ([16]).

Lemma 3. Given an ECTL$^+$ formula G, *if* SNF$_{\mathrm{CTL}}(G)$ is satisfiable then so is G.

Proof. From Lemma 1 it follows that given an ECTL$^+$ formula G, G is satisfiable if, and only if, $\tau_1(G)$ is satisfiable. Thus, for the proof of Lemma 3 we must show that the following proposition takes place:

Proposition 5. Given an ECTL$^+$ formula G, if $\tau_2(\tau_1(G))$ is satisfiable then so is $\tau_1(G)$.

Here we sketch the proof for the new core technique introduced in our transformation procedure. We will show that given $(\dagger)$ $\mathbf{A}\,\Box\,(P \Rightarrow \mathbf{A}(\beta_1 \vee \ldots \vee \beta_m))$ and having generated

$$\left.\begin{array}{ll} (a_1) & \mathbf{A}\,\Box\,(P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{\langle LC(\mathsf{ind_1})\rangle} \\ \ldots & \\ (a_n) & \mathbf{A}\,\Box\,(P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{\langle LC(\mathsf{ind_n})\rangle} \end{array}\right. \quad (\ddagger)$$

(where at least one of β_i $(1 \leqslant i \leqslant m)$ has a form of $\Diamond\,\Box\,Q$ or $\Box\,\Diamond Q$), if $(\ddagger)$ is satisfiable then $(\dagger)$ is satisfiable.

Proof. Consider a model $\mathcal{M}$ which satisfies $(\ddagger)$. We have $\langle \mathcal{M}, s_0\rangle \models (a_1) \wedge \ldots \wedge (a_n)$. Following [16], we know that if a formula with n path quantifiers has a model, then it has an $(n+1)$'ary canonical model. We will now construct this canonical model $\mathcal{M}'$ and show that every state in the model also satisfies $\dagger$. The construction proceeds by first selecting a path from $\mathcal{M}$ which satisfies one of the a_i $(1 \leqslant i \leqslant n)$ clauses, say φ. This will be a basis path to construct a canonical model, which is also referred to as the 'leftmost' path of the canonical model in [16]. Due to the labelling of the states of this path, each of them satisfies $P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{\langle LC(\mathsf{ind_i})\rangle}$. Then inductively construct each of the n additional paths (corresponding to n *somepath* quantifiers) from each state along φ.

Again, we label the states of these paths based on the original interpretations from $\mathcal{M}$ such that each of them also satisfies $P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{\langle LC(\mathsf{ind_i})\rangle}$ (for some i). We then proceed in the same way to take each state of the newly constructed paths and generate the n additional paths from each of them to derive the completed canonical model.

Our ultimate task is to show that for any state in the canonical model $\mathcal{M}'$ which satisfies P, every path emanating from it satisfies $\beta_1 \vee \ldots \vee \beta_m$. This will ensure that $P \Rightarrow \mathbf{A}(\beta_1 \vee \ldots \vee \beta_m))$ is satisfied at every state of $\mathcal{M}'$, and, therefore, it is satisfied in the root of $\mathcal{M}'$. Consider an arbitrarily chosen path χ_i of $\mathcal{M}'$ and a state $s_k \in \chi_1$, see Figure 4. By the construction of $\mathcal{M}'$, every one of the n paths emanating from s_k satisfies $\beta_1 \vee \ldots \vee \beta_m$. What is left is to show that $Suf(\chi_i, s_k)$ (which corresponds to the $n+1$ path emanating from s_k) also satisfies $\beta_1 \vee \ldots \vee \beta_m$. The latter follows from the labelling of the states of the path χ_i which is taken from one of the paths of $\mathcal{M}$ that satisfies one of the ($a_1, \ldots, a_n$).

7. Complexity

Given an arbitrary formula, W, we present the maximum number of $\mathrm{SNF_{CTL}}$ clauses, $clauses(\tau(\mathbf{A}\,\Box(W))$, generated after applying the translation procedure, and also the maximum number of new propositions generated by the translation procedure given by $props(\tau(\mathbf{A}\,\Box(W))$. Our proofs follow the format of analogous proofs of the complexity for the linear-time case [14].

We now define the length *'len'* of an ECTL^{+} formula. In the following formulae $\mathbf{P}$ refers to either of $\mathbf{A}$ or $\mathbf{E}$ quantifiers, l is a literal, n is the number of existential

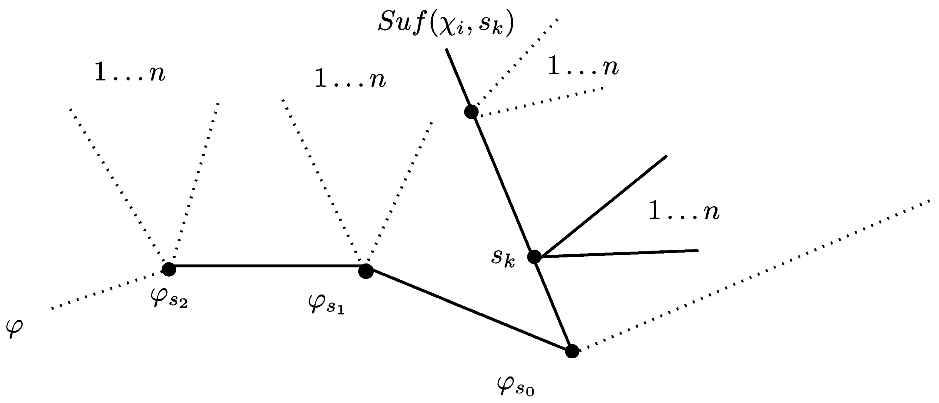

Figure 4. $(n+1)$'ary Canonical Model for ‡.

quantifiers in the original formula, m is the number of elements in an $\mathbf{A}(B_1 \vee \ldots \vee B_m)$ clause, and **const** is **true**, ¬**true**, **false** or ¬**false**.

$$\begin{array}{rcl}
len(\mathbf{P}\Diamond l) & = & 1;\\
len(l_1 \vee l_2 \vee \ldots \vee l_n) & = & 1 \text{ for}(n \geqslant 1);\\
len(\mathbf{const}) & = & 1;\\
len(\mathbf{P}\bigcirc(l_1 \vee l_2 \vee \ldots \vee l_n)) & = & 1 \text{ for}(n \geqslant 1);\\
len(\mathbf{P}\ \Box\ A) & = & len(\mathbf{P}\Diamond A) = len(\mathbf{P}\bigcirc A) = 1 + len(A);\\
len(\neg\mathbf{P}\ \Box\ A) & = & len(\neg\mathbf{P}\Diamond A) = len(\neg\mathbf{P}\bigcirc A) = \\
 & & 1 + len(\neg A);\\
len(A \vee B) = len(A \wedge B) & = & len(\mathbf{P}(A\mathcal{U}\ B)) = len(\mathbf{P}(A\,\mathcal{W}\,B)) = 1 + len(A) + len(B);\\
len(\neg(A \vee B)) & = & len(\neg(A \wedge B)) = len(\neg\mathbf{P}(A\mathcal{U}B)) = \\
 & & len(\neg\mathbf{P}(A\,\mathcal{W}\ B)) = 1 + len(\neg A) + len(\neg B);\\
len(\neg(A \Rightarrow B) & = & 1 + len(A) + len(\neg B);\\
len(A \Rightarrow B) & = & 1 + len(\neg A) + len(B);\\
len(\mathbf{P}A) & = & 1 + len(A);\\
len(\neg\mathbf{P}A) & = & 1 + len(\neg A).
\end{array}$$

7.1. Number of clauses generated

Theorem 2 (New SNF$_{\mathbf{CTL}}$ clauses). For any proposition symbol x and ECTL$_+$ formula W, the maximum number of SNF$_{CTL}$ clauses generated from the translation of $\tau[\mathbf{A}\ \Box\ (x \Rightarrow W)]$, denoted by clauses $(\tau[\mathbf{A}\Box(x \Rightarrow W)])$, will be at most $(12\times\ m \times n^2)$, i.e.,

$$clauses(\tau[\mathbf{A}\Box(x \Rightarrow W)]) \leqslant (12 \times m \times n^2 \times len(W)), (m, n \geqslant 1)$$

Proof. The proof is by induction on the length of W. The base case is where W has length 1, i.e., it has the form $\mathbf{P}\Diamond l, l_1 \vee \ldots \vee l_m$, *true*, *false* or $\mathbf{P}\bigcirc l_1 \vee \ldots \vee l_m$). As illustrated in section 5.2, $\tau[\mathbf{A}\ \Box\ (x \Rightarrow \mathbf{P}\Diamond l)]$ produces one SNF$_{CTL}$ clause, $\tau[\mathbf{A}\ \Box\ (x \Rightarrow (l_1 \vee l_2 \vee \ldots \vee l_n)]$ produces two SNF $_{CTL}$ clauses, and $\tau[\mathbf{A}\ \Box\ (x \Rightarrow const)]$ produces two SNF$_{CTL}$ clauses, and $\tau[\mathbf{A}\ \Box\ (x \Rightarrow \mathbf{P}\bigcirc(l_1 \vee l_2 \vee \ldots \vee l_n)]$ produces one SNF$_{CTL}$ clause. In each case if the number of SNF$_{CTL}$ clauses is M,

$$M \leqslant (12 \times m \times n^2), (m, n \geqslant 1)$$

For the inductive hypothesis we assume that the theorem holds for formulae of length q and examine each case for length $q + 1$. Considering the transformations in section 5.2, if the maximum number of SNF $_{CTL}$ clauses from removing any operator (or negated operator) is M then $M \leqslant (m \times n^2 \times 12)$ which occurs in the $\mathbf{A}(B_1 \vee B_2 \vee$

$\ldots \vee B_m)$ case where each $B_i (i \leqslant m)$ is of the form $(\mathbf{A}\Diamond \Box\, C)$, and each C of the form $\neg\mathbf{P}(C_1\, \mathcal{W}\, C_2)$.

For each $B_i (i \leqslant m)$ of the form $(\mathbf{A}\Diamond \Box\, C)$, since each $B_i (i \leqslant m) \leqslant 11$, given by the case $\neg P(C_1\, \mathcal{W}\, C_2)$ (see below) we have

$$\begin{aligned} clauses(\tau[\mathbf{A}\,\Box\,(x \Rightarrow \mathbf{A}(B_1 \vee B_2 \vee \ldots \vee B_m)]) &= n \times [(n \times (1 + len(\mathbf{E}\,\Box\,\mathbf{B_1}))+ \\ &\quad \ldots + (n \times (1 + len(\mathbf{E}\Box\mathbf{B_m}))] \end{aligned}$$

Therefore,

$$clauses(\tau[\mathbf{A}]\,\Box\,(x \Rightarrow \mathbf{A}(B_1 \vee B_2 \vee \ldots \vee B_m)]) \leqslant (n \times m \times [n \times 12])$$

Hence

$$clauses(\tau[\mathbf{A}\,\Box\,(x \Rightarrow \mathbf{A}(B_1 \vee B_2 \vee \ldots \vee B_m)]) \leqslant (m \times n^2 \times 12).$$

Now we provide the proof for $\neg P(C_1\, \mathcal{W}\, C_2)$.

$$\begin{aligned} clauses(\tau[\mathbf{A}\,\Box\,(x \Rightarrow \neg P(C_1\, \mathcal{W}\, C_2))]) &= 11 + clauses(\tau[\mathbf{A}\,\Box\,(y \Rightarrow \neg C_1)]) + \\ &\quad clauses(\tau[\mathbf{A}\,\Box\,(z \Rightarrow \neg C_2)]). \end{aligned}$$

Hence,

$$\begin{aligned} clauses(\tau[\mathbf{A}\,\Box\,(x \Rightarrow \neg P(C_1\, \mathcal{W}\, C_2))]) &\leqslant (11 + (11 \times len(\neg C_1))+ \\ &\quad (11 \times len(\neg C_2))) = \\ &\quad 11(1 + len(\neg C1) + len(\neg C_2)) = \\ &\quad 11 \times len(\neg(C_1\, \mathcal{W}\, C_2)). \end{aligned}$$

The cases for the other operators are similar.

Theorem 3 (Maximum number of SNF$_{\mathbf{CTL}}$ clauses). For any ECTL$^+$ formula W, the maximum number of SNF$_{\mathrm{CTL}}$ clauses generated from the translation into SNF$_{\mathrm{CTL}}$ will be at most $1 + (m \times n^2 \times 12 \times len(W))$.

Proof. Let W be a ECTL$^+$ formula. To transform it into SNF$_{\mathrm{CTL}}$ we apply the τ transformations

$$\tau[W] = \tau[\mathbf{A}\,\Box\,(x \Rightarrow W)] \wedge \mathbf{A}\,\Box\,(start \Rightarrow x)$$

From Theorem 2 above, we know the maximum number of SNF$_{\mathrm{CTL}}$ clauses from $\tau[\mathbf{A}\,\Box\,(x \Rightarrow W)]$ is $\leqslant (m \times n^2 \times 12)$; hence, the maximum number for the translation of W is $1 + (m \times n^2 \times 12) \times len(W)$. $\Box$

Hence we conclude that the transformation procedure is polynomial on the length of W and hence is in P.

7.2. Number of new proposition symbols generated

Theorem 4 (New propositions). For any proposition symbol x and ECTL$^+$ formula, W, the maximum number of new proposition symbols generated from the translation of $\tau[\mathbf{A} \,\square\, (x \Rightarrow W)]$, denoted by $props(\tau[\mathbf{A}\square(x \Rightarrow W)])$, will be at most $(m \times n^2 \times 5 \times \, len\, (W))$, i.e.,

$$props(\tau[\mathbf{A} \,\square\, (x \Rightarrow W)]) \leqslant (5 \times m \times n^2 \times len(W)), for(m, n \geqslant 1)$$

Proof. The proof is by induction on the length of W. The base case is where W has length 1, i.e., it has the form $\mathbf{P}\diamondsuit l, l_1 \vee \ldots \vee l_m, true,\ false$ or $\mathbf{P} \bigcirc (l_1 \vee \ldots \vee l_m)$. Each of these produces no new proposition symbols so as $0 \leqslant 5 \times 1$ we are done. For the inductive hypothesis we assume that the theorem holds for formulae of length q and examine each case for length $q + 1$. We note that the maximum number of propositions added in the translation of a CTL formula is 4 which arises from the translation of $\neg\mathbf{P}(A \,\mathcal{W}\, B)$. When this occurs as C in $(\mathbf{A}\diamondsuit \square C)$ then we add 1 for the translation of each $\mathbf{E}\diamondsuit \square \mathbf{C}$.

For each $B_i (i \leqslant m)$ of the form $(\mathbf{A}\diamondsuit \square C)$, and C of the form $\neg\mathbf{P}(C_1 \,\mathcal{W}\, C_2)$

$$\begin{aligned} props(\tau[\mathbf{A}\square(x \Rightarrow \mathbf{A}(B_1 \vee B_2 \vee \ldots \vee B_m)])) \;=\;\; & n \times [(n \times (1 + props(\mathbf{E}\square\mathbf{B_1})) + \\ & \ldots + (n \times (1 + props(\mathbf{E}\square\mathbf{B_m}))] \end{aligned}$$

Since the number of props in each $B_i (i \leqslant m) \leqslant 4$,

$$props(\tau[\mathbf{A}\square(x \Rightarrow \mathbf{A}(B_1 \vee B_2 \vee \ldots \vee B_m)])) \leqslant (m \times n^2 \times 5).$$

The cases for the other operators are similar. $\square$

Theorem 5 (Maximum number of SNF$_{\mathbf{CTL}}$ propositions). For any ECTL$^+$ formula W, the maximum number of new propositions generated from the translation into SNF$_{CTL}$ will be at most $1 + (m \times n^2 \times 5 \times len(W))$.

Proof. Let W be a ECTL$^+$ formula. To transform it into SNF$_{CTL}$ we apply the τ transformation

$$\tau[W] = \tau[\mathbf{A} \,\square(x \Rightarrow W)] \wedge \mathbf{A} \,\square(start \Rightarrow x)$$

Let N be the maximum number of new proposition symbols, generated from $\tau[\mathbf{A} \,\square(x \Rightarrow W)]$. From Theorem 5, $N \leqslant (m \times n^2 \times 5 \times len(W))$; hence, the maximum number for the translation of W is $\leqslant (m \times n^2 \times 5 \times len(W))$. Hence the maximum number of new propositions from the translation of W is $1 + (m \times n^2 \times 5 \times \, len(W))$. $\square$

8. The temporal resolution method

Having provided the translation of ECTL$^+$ formulae into SNF$_{\mathrm{CTL}}$, we represent all temporal statements within ECTL$^+$ as sets of clauses. Now, in order to achieve a refutation, we incorporate two types of resolution rules already defined in [2, 6]: *step* resolution (SRES) and *temporal* resolution (TRES).

Step resolution rules. Step resolution is used between formulae that refer to the *same* initial moment of time or *same* next moment along some or all paths. In the formulation of the SRES rules below l is a literal and C and D are disjunctions of literals.

SRES 1

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

SRES 2

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

SRES 3

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

SRES 4

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

When an empty constraint is generated on the right hand side of the conclusion of the resolution rule, we introduce a constant **false** to indicate this situation and, for example, the conclusion of the SRES 1 rule, when resolving **start** $\Rightarrow l$ and **start** $\Rightarrow \neg l$, will be **start** $\Rightarrow$ **false**, which is the terminating clause.

Temporal resolution rules. In the rules below l is a literal and the first premises in the TRES rules abbreviate the **A** and **E** loops in l ([5]), i.e., the situation where, given that P is satisfied at some point of time, l occurs always from that point on all or some path, respectively.

TRES 1

$$\frac{\begin{array}{l}P \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\square\, l\\ Q \Rightarrow \mathbf{A}\diamondsuit\neg l\end{array}}{Q \Rightarrow \mathbf{A}(\neg P\,\mathcal{W}\,\neg l)}$$

TRES 2

$$\frac{\begin{array}{l}P \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\square\, l\\ Q \Rightarrow \mathbf{E}\diamondsuit\neg l_{\langle LC(\mathsf{ind})\rangle}\end{array}}{Q \Rightarrow \mathbf{E}(\neg P\,\mathcal{W}\,\neg l)_{\langle LC(\mathsf{ind})\rangle}}$$

TRES 3

$$\frac{\begin{array}{l}P \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\square\, l_{\langle LC(\mathsf{ind})\rangle}\\ Q \Rightarrow \mathbf{A}\diamondsuit\neg l\end{array}}{Q \Rightarrow \mathbf{A}(\neg P\,\mathcal{W}\,\neg l)}$$

TRES 4

$$\frac{\begin{array}{l}P \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\square\, l_{\langle LC(\mathsf{ind})\rangle}\\ Q \Rightarrow \mathbf{E}\diamondsuit\neg l_{\langle LC(\mathsf{ind})\rangle}\end{array}}{Q \Rightarrow \mathbf{E}(\neg P\,\mathcal{W}\,\neg l)_{\langle LC(\mathsf{ind})\rangle}}$$

Correctness of the transformation of ECTL$^+$ formulae into SNF$_{\mathrm{CTL}}$ (section 6) together with the termination and correctness of the resolution method defined over SNF$_{\mathrm{CTL}}$ (shown in [2, 6]) enables us to apply the latter as the refutation method for ECTL$^+$.

Example refutation. We apply the resolution method to the set of SNF$_{\mathrm{CTL}}$ clauses obtained for the ECTL$^+$ formula $\mathbf{A}(\square\Diamond p \wedge \Diamond\square\neg p)$ (Formula (21) in section 5.3). We commence the resolution proof presenting at steps 1–13 only those clauses that are involved in the resolution refutation in the following order: initial clauses, step clauses and, finally, any sometime clauses.

1. $\mathbf{start} \Rightarrow x$
2. $\mathbf{start} \Rightarrow \neg x \vee y$
3. $\mathbf{start} \Rightarrow \neg x \vee x_1$
4. $\mathbf{start} \Rightarrow \neg z \vee \neg p$
5. $\mathbf{start} \Rightarrow \neg z \vee z_1$
6. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg z \vee \neg p)$
7. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg z \vee z_1)$
8. $x_1 \Rightarrow \mathbf{A}\bigcirc y$
9. $x_1 \Rightarrow \mathbf{A}\bigcirc x_1$
10. $z_1 \Rightarrow \mathbf{E}\bigcirc\neg p_{\langle \mathsf{f}\rangle}$
11. $z_1 \Rightarrow \mathbf{E}\bigcirc z_{1\langle \mathsf{f}\rangle}$
12. $y \Rightarrow \mathbf{A}\Diamond p$
13. $x \Rightarrow \mathbf{E}\Diamond z_{\langle LC(\mathsf{f})\rangle}$

We apply step resolution rules between 1 and 2, and 1 and 3. No more SRES rules are applicable. Formula 12 is an eventuality clause, and therefore, we are looking for a loop in $\neg p$ (see [5] for the formulation of the loop searching procedure). The desired loop, $\mathbf{E}\,\square\,\mathbf{E}\bigcirc\neg p_{\langle LC(\mathsf{f})\rangle}$ (given that condition z_1 is satisfied) can be found considering clauses 10 and 11. Thus, we apply the TRES 3 rule to resolve this loop and clause 12, obtaining 16. Next we remove $\mathbf{E}\,\mathcal{W}$ from 16 deriving a purely classical formula 17 (y is a new variable). Simplify the latter, apply temporising, obtaining, in particular, 19 and 20, and then a series of SRES rules to newly generated clauses.

14. $\mathbf{start} \Rightarrow y$	$1, 2,$ *SRES*
15. $\mathbf{start} \Rightarrow x_1$	$1, 3,$ *SRES*
16. $y \Rightarrow \mathbf{A}(\neg z_1\,\mathcal{W} p)$	$10, 11, 12$ *TRES 3*
17. $y \Rightarrow p \vee \neg z_1 \wedge v$	$16,$ $\mathbf{A}\,\mathcal{W}$ Removal
18. $v \Rightarrow \mathbf{A}\bigcirc(p \vee \neg z_1 \wedge v)$	$16,$ $\mathbf{A}\,\mathcal{W}$ Removal
19. $\mathbf{start} \Rightarrow \neg y \vee p \vee \neg z_1$	$17,$ SIMP, TEMP
20. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg y \vee p \vee \neg z_1)$	$17,$ SIMP, TEMP
21. $\mathbf{start} \Rightarrow p \vee \neg z_1$	$14, 19,$ *SRES* 1
22. $\mathbf{start} \Rightarrow p \vee \neg z$	$5, 21,$ *SRES* 1
23. $\mathbf{start} \Rightarrow \neg z$	$4, 22,$ *SRES* 1
24. $x_1 \Rightarrow \mathbf{A}\bigcirc(p \vee \neg z_1)$	$8, 20,$ *SRES* 3
25. $x_1 \Rightarrow \mathbf{A}\bigcirc(p \vee \neg z)$	$7, 24,$ *SRES* 3
26. $x_1 \Rightarrow \mathbf{A}\bigcirc\neg z$	$6, 25,$ *SRES* 3

Now, as no more SRES rules are applicable, we find another eventuality, formula 13, and thus we next look for a loop in $\neg z$. This loop can be found considering formulae 9 and 26: $\mathbf{A} \bigcirc \mathbf{A} \square \neg z$ given that condition x_1 is satisfied. Thus, we can apply TRES 2 to resolve this loop and 13 deriving 27. Then we remove $\mathbf{E}\mathcal{W}$ from the latter (on step 28, where w is a new variable, we use only one of its conclusions). Applying simplification and temporising to 28 we obtain 29. The desired terminating clause **start** $\Rightarrow$ **false** is deduced by applying SRES 1 to steps 1, 15 and 23.

27.	$x \Rightarrow \mathbf{E}(\neg x_1 \, \mathcal{W} z)_{\langle LC(\mathsf{f})\rangle}$	$9, 26, 13$ *TRES* 2
28.	$x \Rightarrow z \vee \neg x_1 \wedge w$	27 $\mathbf{E}\mathcal{W}$ Removal
29.	$\mathbf{start} \Rightarrow \neg x \vee z \vee \neg x_1$	28 SIMP, TEMP
30.	$\mathbf{start} \Rightarrow \mathbf{false}$	$1, 15, 23$ *SRES* 1

9. Conclusions and future work

We have described the extension of the clausal resolution method to the useful branching-time logic ECTL$^+$. Here we have followed our general idea to expand the applicability of the clausal resolution technique originally developed for linear-time temporal logic [13], and further extended to branching-time temporal logics CTL and ECTL [2, 3, 6]. This extension enables us to invoke a variety of well-developed methods and refinements used in the resolution framework for classical logic (see, for example, [1]). The algorithm to search for loops needed for temporal resolution has been introduced in [5]. With the proof that $\mathrm{SNF}_{\mathrm{CTL}}$ can be served as the normal form for ECTL$^+$, the algorithm becomes fully functional for the latter. Another contribution of this paper is providing the complexity analysis of the transformation of ECTL$^+$ formulae into $\mathrm{SNF}_{\mathrm{CTL}}$, namely, we have now shown that the complexity of the transformation procedure is polynomial in the length of the original ECTL$^+$ formula. Our results have brought us one step closer to the final stage of our long-term project – to define a clausal resolution method for CTL*. Among other obvious tasks are to *refine* the presented method and to analyse the complexity of the resolution method in whole which would enable the development of the corresponding prototype systems.

References

[1] L. Bachmair and H. Ganzinger, A theory of resolution, in: *Handbook of Automated Reasoning*, eds. J.A. Robinson and A. Voronkov, chapter 2 (Elsevier, 2001).

[2] A. Bolotov, *Clausal Resolution for Branching-Time Temporal Logic*, PhD thesis, Department of Computing and Mathematics, The Manchester Metropolitan University, 2000.

[3] A. Bolotov, Clausal resolution for extended computation tree logic ECTL, in: *Proceedings of the Time – 2003/International Conference on Temporal Logic 2003* (Cairns, IEEE, July 2003).

[4] A. Bolotov and A. Basukoski, Clausal resolution for extended computation tree logic ECTL. *Journal of Applied Logic* (In Press).

[5] A. Bolotov and C. Dixon, Resolution for branching time temporal logics: Applying the temporal resolution rule, in: *Proceedings of the 7th International Conference on Temporal Representation and Reasoning (TIME2000)* pp. 163–172, Cape Breton, Nova Scotia, Canada, 2000. IEEE Computer Society.

[6] A. Bolotov and M. Fisher, A clausal resolution method for CTL branching time temporal logic, Journal of Experimental and Theoretical Artificial Intelligence 11 (1999) 77–93.

[7] J. Bradfield and C. Stirling, Modal logics and mu-calculi, in: *Handbook of Process Algebra*, eds. J. Bergstra, A. Ponse and S. Smolka (Elsevier, North-Holland, 2001) pp. 293–330.

[8] E.A. Emerson, Temporal and modal logic, in: *Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics,* eds. J. van Leeuwen (Elsevier, 1990) pp. 996–1072.

[9] E.A. Emerson, Automated reasoning about reactive systems, in: *Logics for Concurrency: Structures Versus Automata, Proc. of International Workshop*, vol. 1043 of Lecture Notes in Computer Science (Springer, 1996) pp. 41–101.

[10] E.A. Emerson and J.Y. Halpern, Decision procedures and expressiveness in the temporal logic of branching time JCSS 30(1) (1985) 1–24.

[11] E.A. Emerson and J.Y. Halpern, "Sometimes'' and "Not never'' revisited: On branching versus linear time temporal logic, *JACM* 33(1) (1986) 151–178.

[12] E.A. Emerson and A.P. Sistla, Deciding full branching time logic, in: *STOC 1984, Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 14–24, 1984.

[13] M. Fisher, A resolution method for temporal logic, in: *Proc. of the XII International Joint Conference on Artificial Intelligence (IJCAI)* (1991) pp. 99–104.

[14] M. Fisher, C. Dixon and M. Peim, Clausal temporal resolution, ACM Transactions on Computational Logic (TOCL) 1(2) (2001) 12–56.

[15] T. Hafer and W. Thomas, Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree, in: *Automata, Languages and Programming, Proc. 14 ICALP*, vol. 267 of *Lecture Notes in Computer Science* (1987) pp. 269–279.

[16] P. Wolper, On the relation of programs and computations to models of temporal logic, in: *Time and Logic, a computational approach,* eds. L. Bolc and A. Szałas, chapter 3 (UCL, 1995) pp. 131–178.