A clausal resolution method for branching-time logic ECTL.

Alexander Bolotov
Artie Basukoski

Harrow School of Computer Science

A Clausal Resolution Method for Branching-Time Logic ECTL$^+$

Alexander Bolotov and Artie Basukoski
Harrow School of Computer Science,
University of Westminster, HA1 3TP, UK
{A.Bolotov,A.Basukoski@wmin.ac.uk}

Abstract

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 that any ECTL$^+$ formula can be translated to a normal form the structure of which was initially defined for CTL and then applied to ECTL. This enables us to apply to ECTL$^+$ a resolution technique defined over the set of clauses. Our correctness argument also bridges the gap in the correctness proof for ECTL: we show that the transformation procedure for ECTL preserves unsatisfiability.

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 [7]. Two combinations of future time temporal operators $\diamondsuit$ ('sometime') and $\square$ ('always'), are useful in expressing *fairness* [6]: $\diamondsuit\square p$ (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 [9]) 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 [2] 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 an ECTL formula 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 latter also bridges the gap in the correctness proof for ECTL: we show that the transformation procedure for ECTL preserves unsatisfiability.

The structure of the paper is as follows. In §2 we outline the syntax and semantics of ECTL$^+$ and those properties that are important for our analysis. In §3 we review SNF$_{\mathrm{CTL}}$. Next, in §4, 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 core of this paper, the proof of the correctness of this transformation technique, is given in §5. Further, in §6 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 §4.3). Finally, in §7, we draw conclusions and discuss future work.

2. Syntax and Semantics of ECTL$^+$

In the language of ECTL$^+$ we extend the language of linear-time temporal logic, which uses future time $\square$ (always), $\diamondsuit$ (sometime), $\bigcirc$ (next time), $\mathcal{U}$ (until) and $\mathcal{W}$ (unless), by path quantifiers **A** (on all future paths) and **E** (on some future path). 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)

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

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

We interpret a well-formed ECTL$^+$ formula in a tree-like model structure $\mathcal{M} = \langle S, R, L\rangle$, where S is a set of states, $R \subseteq S \times S$ is a binary relation over S, and L is an interpretation function mapping atomic propositional symbols to truth values at each state. A path, χ_{s_i}, over R, is a sequence of states $s_i, s_{i+1}, s_{i+2} \ldots$ such that for all $j \geq i$, $(s_j, s_{j+1}) \in R$. A path χ_{s_0} is called a fullpath. 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 $Suf(\chi_{s_i}, s_j)$.

We assume that an ECTL$^+$ model $\mathcal{M}$ satisfies the following conditions: (i) There is a designated state, $s_0 \in S$, a root of a structure (i.e. for all j, $(s_j, s_0) \notin R$); (ii) Every state belongs to some fullpath and should have a successor state; (iii) Tree structures are of at most countable branching; (iv) Every path is isomorphic to ω.

When trees are considered as models for distributed systems, paths through a tree are viewed as computations. The requirements for ECTL$^+$ models we are interested in would be suffix, fusion and limit closures [6].

Below, we define a relation '$\models$', which evaluates well-formed ECTL$^+$ formulae at a state s_i in a model $\mathcal{M}$ omitting standard cases for Booleans.

$\langle\mathcal{M}, s_i\rangle \models p$ iff $p \in L(s_i)$, for atomic p.
$\langle\mathcal{M}, s_i\rangle \models \mathbf{A}B$ iff for each χ_{s_i}, $\langle\mathcal{M}, \chi_{s_i}\rangle \models B$.
$\langle\mathcal{M}, s_i\rangle \models \mathbf{E}B$ iff for some χ_{s_i}, $\langle\mathcal{M}, \chi_{s_i}\rangle \models B$.
$\langle\mathcal{M}, \chi_{s_i}\rangle \models A$ iff $\langle\mathcal{M}, s_i\rangle \models A$, for state formula A.
$\langle\mathcal{M}, \chi_{s_i}\rangle \models \Box B$ iff for each $s_j \in \chi_{s_i}$, if $i \leq j$ then $\langle\mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$.
$\langle\mathcal{M}, \chi_{s_i}\rangle \models \Diamond B$ iff for some $s_j \in \chi_{s_i}, (i \leq j)$ $\langle\mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$.
$\langle\mathcal{M}, \chi_{s_i}\rangle \models \bigcirc B$ iff $\langle\mathcal{M}, Suf(\chi_{s_i}, s_{i+1})\rangle \models B$.
$\langle\mathcal{M}, \chi_{s_i}\rangle \models A\,\mathcal{U}\,B$ iff for some $s_j \in \chi_{s_i} (i \leq j)$ $\langle\mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$ and for each $s_k \in \chi_{s_i}$, if $i \leq k < j$ then $\langle\mathcal{M}, Suf(\chi_{s_i}, s_k)\rangle \models A$.
$\langle\mathcal{M}, \chi_{s_i}\rangle \models A\,\mathcal{W}\,B$ iff $\langle\mathcal{M}, \chi_{s_i}\rangle \models \Box A$ or $\langle\mathcal{M}, \chi_{s_i}\rangle \models A\,\mathcal{U}\,B$.

Definition 1 *An ECTL$^+$ formula B is satisfiable if, and only if, there exists a model $\mathcal{M}$ such that $\langle\mathcal{M}, s_0\rangle \models B$. Formula B is valid if, and only if, every model satisfies it.*

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

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

which will be served in our example of the transformation towards SNF$_{\mathrm{CTL}}$ in §4.3. It is straightforward from the semantics, that this formula is unsatisfiable: take an arbitrary fullpath, say φ, and show that $\Box\Diamond p \wedge \Diamond\Box\neg p$ can not be satisfied along φ as in the linear-time logic.

2.1. 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* $\mathrm{NNF}_{\mathrm{ECTL}^+}(G)$, $\langle\mathcal{M}, s_0\rangle \models G$ iff $\langle\mathcal{M}, s_0\rangle \models \mathrm{NNF}_{\mathrm{ECTL}^+}(G)$ [6].

Given a CTL formula F, we will abbreviate the expression 'a state subformula F_i with a path quantifier as its main operator' by **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 2 (Degree of path quantifier nesting)

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

Emerson and Sistla [10] showed that for any CTL* (hence ECTL$^+$) formula F with $N(F) > 2$, F can be transformed into F' by a continuous renaming of the **P**-embedded state subformulae 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 $Red[G] = \mathbf{A}\Box(x_1 \equiv \mathbf{E}\Box p) \wedge \mathbf{A}\Box(x_2 \equiv \mathbf{E}\Diamond x_1) \wedge \mathbf{A}\Box(x_3 \equiv \mathbf{E}(\Box\Diamond\neg p \wedge \bigcirc q)) \wedge \mathbf{A}\Diamond(x_3 \vee x_2)$

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 $\langle\mathcal{M}'\rangle$ such that $\langle\mathcal{M}', s_0\rangle \models Red(C)$, where Red is introduced in Definition 2* [10].

Recall that the logic CTL$^+$ extends CTL by allowing Boolean combinations of temporal operators (but not any nesting of them). Yet, it is still as expressive as CTL [8]. Hence we can transform any ECTL$^+$ formula which is also a formula of CTL$^+$ into an equivalent CTL formula. Here we give some of the equivalences used for such reduction, referring the reader to the mentioned paper for other cases which involve the $\mathcal{U}$ and $\mathcal{W}$ operation. (In the formulae below B and C are purely classical expressions.)

$$\begin{array}{ll} (a) & \mathbf{P}(\bigcirc B \wedge \bigcirc C) \equiv \mathbf{P}\bigcirc(B \wedge C) \\ (b) & \mathbf{P}(\bigcirc B \vee \bigcirc C) \equiv \mathbf{P}\bigcirc(B \vee C) \\ (c) & \mathbf{P}(\Box B \wedge \Box C) \equiv \mathbf{P}\Box(B \wedge C) \\ (d) & \mathbf{P}(\Diamond B \vee \Diamond C) \equiv \mathbf{P}\Diamond(B \vee C) \\ (e) & \mathbf{E}(B \vee C) \equiv \mathbf{E}B \vee \mathbf{E}C \\ (f) & \mathbf{A}(B \wedge C) \equiv \mathbf{A}B \wedge \mathbf{A}C \end{array} \quad (2)$$

Like ECTL, ECTL$^+$ allows limited nesting of temporal operators to express fairness constraints. For some of them, namely, for $\mathbf{A}\Box\Diamond$ and $\mathbf{E}\Diamond\Box$ cases, the validity of the following equivalences which we will use in our transformation procedure can be easily shown:

$$\begin{array}{l} \text{(a) } \mathbf{A}\Box\Diamond B \equiv \mathbf{A}\Box\mathbf{A}\Diamond B \\ \text{(b) } \mathbf{E}\Diamond\Box B \equiv \mathbf{E}\Diamond\mathbf{E}\Box B \end{array} \quad (3)$$

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 3 (*$DNF_{\mathbf{E}}$* and *$CNF_{\mathbf{A}}$* for ECTL$^+$ formulae) *Let us call formulae of the type $\mathbf{T}(F_1)$, $F_1\mathbf{T}^2F_2$, $\Diamond\Box F_1$, $\Box\Diamond F_1$ (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 α_i $(1 \leq i \leq n)$ is an elementary formula.*

For example, the following formula (which we used in §2 as an unsatisfiability example) $\mathbf{A}(\Box\Diamond p \wedge \Diamond\Box\neg p)$ 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 that has the degree of path quantifiers nesting 1, there exist its $DNF_{\mathbf{E}}(F)$ and $CNF_{\mathbf{A}}(F)$ 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 constraints, $\mathbf{P}\Box\Diamond$ and $\mathbf{P}\Diamond\Box$. Our translation to $\mathrm{SNF}_{\mathrm{CTL}}$ and temporal resolution rules are essentially based upon the fixpoint characterizations of basic CTL modalities (see [5]).

Next we observe some results on interpreting ECTL$^+$ over *canonical models*, noting that these results cover all CTL-type logics, including CTL*.

Definition 4 (Branching degree and branching factor) *The number of immediate successors of a state s in a tree structure is called a* branching degree *of s. Given a set $\mathcal{K} = \{k_1, k_2, \ldots\}$, of the branching degrees of the states of a tree structure, the maximal k_i $(1 \leq i)$ is called a* branching factor *of this tree structure.*

As we have already mentioned, we assume that underlying ECTL$^+$ tree models are of at most countable branching. However, following ([6]), trees with arbitrary, even uncountable, branching, "as far as our branching temporal logics are concerned, are indistinguishable from trees with finite, even bounded, branching".

Definition 5 (Labelled tree) *Given a tree $\mathcal{T} = (N, E)$, where N is a set of nodes and E is a set of edges, and a finite alphabet, Σ, a $\Sigma - labelled$ tree is a structure $(\mathcal{T}, \mathcal{L})$ where $\mathcal{L}$ is a mapping $N \dashrightarrow \Sigma$, which assigns for each state, element of N, some label, element of Σ.*

In §2 we introduced the notion of satisfiability and validity of ECTL$^+$ formulae in relation to $\langle \mathcal{M}, s_0 \rangle$. Now, following [12], we 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 [12], 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 $\{(s, s_i) | s \in [k]^\star,\ i \in \mathcal{K}\}$, where $[k]^\star = N$ and s_i $(i \in \mathcal{K})$ is a set of successors of a state s.

Definition 6 (Tree canonical interpretation) *Let $\mathcal{T} = (N, E)$ be a k-ary infinite tree such that $[k]$ denotes the set $\{1, \ldots k\}$, of branching degrees of the states in $\mathcal{T}$ and $E = \{(s, s_i) | s \in [k]^\star,\ i \in \mathcal{K}\}$. 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 $\mathcal{M} = ([k]^\star, E, \mathcal{L})$ such that $\mathcal{L} : [k]^\star \dashrightarrow 2^{Prop}$.*

In a canonical interpretation $\langle ([k]^\star, E, \mathcal{L}), \lambda \rangle$ the set of states, the initial state and the successor relation are all fixed, hence, "...they reduce to a function $[k]^\star \dashrightarrow 2^{Prop}$, that is to a labelled tree over the alphabet 2^{Prop}..." ([12]). We will refer to this tree as a *canonical model*. Proposition 4 given below collects the results given in [12] (Lemma 3.5).

Proposition 4 (Existence of a canonical model) *If an ECTL$^+$ formula F with n $\mathbf{E}$-quantifiers has a model, then it has an $(n+1)$-ary canonical model.*

These results were essentially used in the formulation of the transformation rule for the ECTL fairness constraint $\mathbf{A}\Diamond\Box$ [2]. In this paper we will further extend their applicability in the transformation procedure for ECTL$^+$.

3. Normal Form for ECTL$^+$

As a normal form for ECTL$^+$, similarly to ECTL, we utilise a clausal normal form, defined for the logic CTL, $\mathrm{SNF}_{\mathrm{CTL}}$, which was developed in [1, 4]. All formulae of $\mathrm{SNF}_{\mathrm{CTL}}$ of the type $P \Rightarrow \mathbf{E}\bigcirc Q$ or $P \Rightarrow \mathbf{E}\Diamond Q$ (see below), where Q is a purely classical expression, are labelled with some index. Indices are used to preserve a specific path context during the translation. 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. A designated type of indices in SNF$_{\mathrm{CTL}}$ are indices of the type $\langle \mathsf{LC(ind)}\rangle$ which represents a limit closure of $\langle \mathsf{ind}\rangle$. Thus, $\mathbf{E}A_{\langle \mathsf{f}\rangle}$ means that A holds on some path labelled as $\langle \mathsf{f}\rangle$, for some f.

The alphabet for SNF$_{\mathrm{CTL}}$ language is obtained from ECTL$^+$ by omitting the $\mathcal{U}$ and $\mathcal{W}$ operators, adding classically defined constants **true** and **false** and a new operator, **start** ('at the initial moment of time') defined as $\langle \mathcal{M}, s_i\rangle \models \mathbf{start}$ iff $i = 0$.

Definition 7 (Separated Normal Form SNF$_{\mathbf{CTL}}$) SNF$_{\mathrm{CTL}}$ *is a set of formulae* $\mathbf{A}\square\,[\bigwedge_i (P_i \Rightarrow F_i)]$ *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
$\bigwedge_{j=1}^{l} \alpha_j$	$\Rightarrow$	$\mathbf{A}\bigcirc\,[\bigvee_{m=1}^{n} \beta_m]$	an **A** step clause
$\bigwedge_{p=1}^{q} \alpha_p$	$\Rightarrow$	$\mathbf{E}\bigcirc\,[\bigvee_{r=1}^{s} \beta_r]_{\langle \mathsf{ind}\rangle}$	a **E** step clause
$\bigwedge_{t=1}^{u} \alpha_t$	$\Rightarrow$	$\mathbf{A}\Diamond\gamma$	an **A** sometime clause
$\bigwedge_{v=1}^{w} \alpha_v$	$\Rightarrow$	$\mathbf{E}\Diamond\gamma_{\langle \mathsf{LC(ind)}\rangle}$	a **E** sometime clause

We obtain the SNF$_{\mathrm{CTL}}$ semantics from the semantics of ECTL$^+$ (§2) by preserving only items for state formulae.

4. Transformation of ECTL$^+$ formulae into SNF$_{\mathbf{CTL}}$

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

4.1. Algorithm to transform ECTL$^+$ formulae into SNF$_{\mathbf{CTL}}$

As SNF$_{\mathrm{CTL}}$ is a part of the resolution technique, to check validity of an ECTL$^+$ formula G, we first negate the latter and translate $\neg G$ into its Negation Normal Form, deriving $C =$NNF$_{\mathrm{ECTL}}(\neg G)$. 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 apply the initial renaming rule obtaining $\mathbf{A}\square(\mathbf{start} \Rightarrow x_0) \wedge \mathbf{A}\square(x_0 \Rightarrow C)$, where x_0 is a new proposition.

(2) Apply equations (3) 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 $\mathbf{PT}C_j, \mathbf{P}\Diamond\square C_j, \mathbf{P}\square\Diamond C_j, \mathbf{P}(C_{j_1}\mathbf{T}^2 C_{j_2}), \mathbf{P}(\mathbf{T}C_{j_1} \wedge \ldots \wedge \mathbf{T}C_{j_n}), \mathbf{P}(\mathbf{T}C_{j_1} \vee \ldots \vee \mathbf{T}C_{j_m})$ (for some $n, m \geq 1$) 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 2-(e) or 2-(f) respectively.

(3.1.2) apply equivalences (3).

(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 2.1).

(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 **PT** is not $\mathbf{P}\bigcirc$ then C_j is a literal, else C_j is a purely classical formula.
- If $Q_j = \mathbf{E}\square\Diamond C_j$ or $Q_j = \mathbf{A}\Diamond\square C_j$ then C_j is a literal,
- If $Q_j = \mathbf{P}(C_{j_1}\mathbf{T}^2 C_{j_2})$ 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 α_k $(1 \leq k \leq n)$ and β_l $(1 \leq l \leq m)$ 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 **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 $\mathbf{E}\square\Diamond$ and $\mathbf{A}\Diamond\square$.

(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$_{\mathrm{CTL}}$ 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 $\mathbf{P}\bigcirc$ ap-

plies either to a literal or to disjunction of literals. This can be achieved, again, by renaming of the embedded classical subformulae, applying rules used to obtain 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.

4.2. Transformation rules towards $\mathrm{SNF}_{\mathrm{CTL}}$

The first stage of the transformation procedure τ outlined above, the procedure τ_1, is taken from the translation of ECTL formulae to $\mathrm{SNF}_{\mathrm{CTL}}$ [2]. Here we describe novel techniques to cope with Boolean combinations of temporal operators defined in addition to the rules of the procedure τ_2, introduced for ECTL [2]. We also recall some of those rules that will be used in our example given in §4.3. For the full set of rules preserved from the CTL the reader is referred to [1, 4].

In the presentation below we omit the outer '$\mathbf{A}\,\square$' connective that surrounds the conjunction of pre-clauses and, for convenience, consider a set of pre-clauses rather than the conjunction. Expressions P and Q will abbreviate purely classical formulae.

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 [1, 2] 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. Here we give those removal rules that will be used in our examples of the transformation to $\mathrm{SNF}_{\mathrm{CTL}}$ (§4.3) and refutation (§6). In the formulation of the rules given below x is a new proposition:

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 $\mathbf{E}\mathcal{W}$

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

Managing embedded path subformulae in ECTL^+. We incorporate rules to rename purely path formulae embedded in ECTL^+ fairness constraints from [2]. Let the number of indices in *LIST_IND* be n ($n \geq 0$) and let $\langle \mathsf{ind_1}\rangle, \ldots \langle \mathsf{ind_n}\rangle \in IND$ be the constants occurring in these indices. If 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: the $\mathbf{E}\,\square\lozenge$ case.

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

Renaming: the $\mathbf{A}\lozenge\,\square$ case.

if $n = 0$

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

if $n > 0$

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

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 $n > 0$

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

Finally, from the rest of the rules previously defined for CTL/ECTL, we use the following.

Temporising

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

Distributivity of $\mathbf{E}\bigcirc$

$$\frac{P \Rightarrow \mathbf{E}\bigcirc(P \wedge Q)_{\langle LC(\mathsf{ind})\rangle}}{\begin{array}{l} P \Rightarrow \mathbf{E}\bigcirc P_{\langle LC(\mathsf{ind})\rangle} \\ P \Rightarrow \mathbf{E}\bigcirc Q_{\langle LC(\mathsf{ind})\rangle}\end{array}}$$

In the rule for $\mathbf{E}\bigcirc$, given that the premise of the rule is labelled by $\langle LC(\mathsf{ind})\rangle$, we preserve this label for both conclusions, thus, assuring that they refer to the same path.

4.3. Example Transformation

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

$$\mathbf{E}(\lozenge\,\square\neg p \vee \square\lozenge p) \qquad (4)$$

To check if (4) is valid negate it and 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 §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 (2)-(f) to distribute the **A** over conjunction in 2, obtaining 3, and then simplify the latter deriving 4 and 5. Next, we simplify formula 4 applying (3-(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$(2-\mathsf{f})$
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$(3-(\mathsf{a}))$
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 (3-(b)) 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 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 **A** and **E** over $\bigcirc$ in 13 and 15.

16.	**start**	$\Rightarrow$	$\neg x \vee y$	12, Simp, Temp
17.	**start**	$\Rightarrow$	$\neg x \vee x_1$	12, Simp, Temp
18.	**true**	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg x \vee y)$	12, Simp, Temp
19.	**true**	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg x \vee x_1)$	12, Simp, Temp
20.	**start**	$\Rightarrow$	$\neg z \vee \neg p$	14, Simp, Temp
21.	**start**	$\Rightarrow$	$\neg z \vee z_1$	14, Simp, Temp
22.	**true**	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg z \vee \neg p)$	14, Simp, Temp
23.	**true**	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg z \vee \neg z_1)$	14, Simp, Temp
24.	x	$\Rightarrow$	$\mathbf{A}\bigcirc y$	13, equiv$(2-\mathsf{a})$
25.	x	$\Rightarrow$	$\mathbf{A}\bigcirc x_1$	13, equiv$(2-\mathsf{a})$
26.	z_1	$\Rightarrow$	$\mathbf{E}\bigcirc\neg p_{\langle\mathsf{f}\rangle}$	15, Dist. $\mathbf{E}\bigcirc$ over $\wedge$
27.	z_1	$\Rightarrow$	$\mathbf{E}\bigcirc z_{1\,\langle\mathsf{f}\rangle}$	15, Dist. $\mathbf{E}\bigcirc$ over $\wedge$

The normal form of the given ECTL$^+$ formula $\mathbf{A}(\Box\Diamond p \wedge \Diamond\Box\neg p)$ is represented by clauses 1, 8, 10, 16–27.

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

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 [1, 2] for CTL and ECTL or extend these proofs for new cases of ECTL$^+$ formulae. Therefore, we will only state such claims referring the reader to [1, 2] while we sketch here proofs for new techniques used for ECTL$^+$ transformations. Note also that in our previous paper ([2]) 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, $\mathrm{SNF}_{\mathrm{CTL}}(G)$, if $\mathrm{SNF}_{\mathrm{CTL}}(G)$ 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 ([2]).

Lemma 2 *Given an* $\mathrm{SNF}_{\mathrm{CTL}}$ *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 ([12]).

Lemma 3 *Given an ECTL$^+$ formula G, if* $\mathrm{SNF}_{\mathrm{CTL}}(G)$ *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 (†) $\mathbf{A}\Box(P \Rightarrow \mathbf{A}(\beta_1 \vee \ldots \vee \beta_m))$ and having generated

$$\begin{array}{lll} (a_1) & \mathbf{A}\Box(P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{\langle LC(\mathsf{ind}_1)\rangle} & \\ \ldots & & (\ddagger) \\ (a_n) & \mathbf{A}\Box(P \Rightarrow \mathbf{E}(\beta_1 \vee \ldots \vee \beta_m)_{\langle LC(\mathsf{ind}_n)\rangle} & \end{array}$$

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

PROOF: Consider a model $\mathcal{M}$ which satisfies (‡). We have $\langle \mathcal{M}, s_0\rangle \models (a_1) \wedge \ldots \wedge (a_n)$. Following [12], 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 †. The construction proceeds by first selecting a path, say φ, from $\langle\mathcal{M}\rangle$ which satisfies one of the a_i $(1 \leq i \leq n)$ clauses. 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 [12]. 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 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.

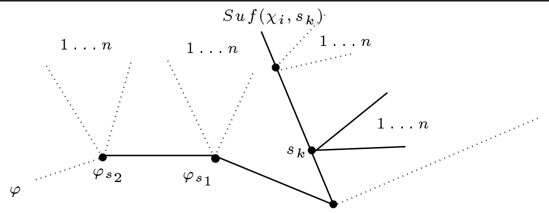

Figure 1. $(n+1)$**'ary Canonical Model for** ‡.

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 1. 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 $(a_1), \ldots, (a_n)$.

6. The Temporal Resolution Method

In order to achieve a refutation, we apply two types of resolution rules already defined in [1, 4]: *step* resolution (SRES) and *temporal* resolution (TRES). The SRES rules are used between formulae which refer to the *same* initial moment of time or *same* next moment along some or all paths. The basic idea of invoking temporal resolution is to resolve a set of formulae characterizing a *loop* in l, i.e. a set of $\mathrm{SNF_{CTL}}$ clauses indicating a situation when l occurs at all future moments along every (an **A**-loop in l) or some path (a **E**-loop in l) from a particular point in an ECTL^+ model, together with the clause containing $\Diamond\neg l$ [3]. Here we present those step (SRES 1 and SRES 2) and temporal resolution (TRES 2 and TRES 4) rules which are used in the example refutation. (For a detailed description of the resolution technique defined over $\mathrm{SNF_{CTL}}$ see [1, 4].)

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

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

TRES 2

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

TRES 4

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

In the rules above l is a literal and the first premises in the TRES rules abbreviate the **A** and **E** loops in l respectively (given that P is satisfied).

Correctness of the transformation of ECTL^+ formulae into $\mathrm{SNF_{CTL}}$ (§5) together with the termination and correctness of the resolution method defined over $\mathrm{SNF_{CTL}}$ (shown in [1, 4]) enables us to apply the latter as the refutation method for ECTL^+.

Example Refutation. We apply the resolution method to the set of $\mathrm{SNF_{CTL}}$ clauses obtained in section §4.3 for the ECTL^+ formula $\mathbf{A}(\Box\Diamond p \wedge \Diamond\Box\neg p)$. 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.	$\mathbf{start} \Rightarrow x$	8.	$x_1 \Rightarrow \mathbf{A}\bigcirc y$
2.	$\mathbf{start} \Rightarrow \neg x \vee y$	9.	$x_1 \Rightarrow \mathbf{A}\bigcirc x_1$
3.	$\mathbf{start} \Rightarrow \neg x \vee x_1$	10.	$z_1 \Rightarrow \mathbf{E}\bigcirc\neg p_{\langle \mathsf{f}\rangle}$
4.	$\mathbf{start} \Rightarrow \neg z \vee \neg p$	11.	$z_1 \Rightarrow \mathbf{E}\bigcirc z_{1\langle \mathsf{f}\rangle}$
5.	$\mathbf{start} \Rightarrow \neg z \vee z_1$	12.	$y \Rightarrow \mathbf{A}\Diamond p$
6.	$\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg z \vee \neg p)$	13.	$x \Rightarrow \mathbf{E}\Diamond z_{\langle LC(\mathsf{f})\rangle}$
7.	$\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg z \vee z_1)$		

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 [3] for the formulation of the loop searching procedure). The desired loop, $\mathbf{E}\,\Box\mathbf{E}\bigcirc\neg p_{\langle LC(\mathsf{f})\rangle}$ (given that condition z_1 is satisfied) can be found considering clauses 10 and 11. Thus, we apply the TRES 4 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.	$y \Rightarrow \mathbf{E}(\neg z_1\,\mathcal{W}\,p)_{\langle LC(\mathsf{f})\rangle}$	$10, 11, 12$ *TRES 4*
17.	$y \Rightarrow p \vee \neg z_1 \wedge v$	$16,\ \mathbf{E}\mathcal{W}$ Removal
18.	$v \Rightarrow \mathbf{E}\bigcirc(p \vee \neg z_1 \wedge v)_{\langle \mathsf{f}\rangle}$	$16,\ \mathbf{E}\mathcal{W}$ 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 are looking for a loop in $\neg z$ which can be found considering formulae 9 and 26: $\mathbf{A}\bigcirc\mathbf{A}\,\Box\neg z$ 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 $\mathbf{start} \Rightarrow \mathbf{false}$ 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.	$\mathbf{start} \Rightarrow \mathbf{false}$	$1, 15, 23\ SRES\ 1$

7. 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 [11], and further extended to branching-time temporal logics CTL and ECTL [4, 1, 2]. This extension enables us to invoke a variety of well-developed methods and refinements used in the resolution framework for classical logic. The algorithm to search for loops needed for temporal resolution has been introduced in [3]. With the proof that 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 completing the proof of the correctness of the transformation procedure in ECTL formulated in [2]: we have now shown that if the set of clauses generated for an ECTL formula is satisfiable then the original formula is satisfiable. 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 its complexity which would enable the development of the corresponding prototype systems.

References

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

[2] A. Bolotov. Clausal resolution for extended computation tree logic ECTL. In *Proceedings of the Time-2003/International Conference on Temporal Logic 2003*, pages 107–117, Cairns, July 2003. IEEE.

[3] 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)*, pages 163–172, Cape Breton, Nova Scotia, Canada, 2000. IEEE Computer Society.

[4] A. Bolotov and M. Fisher. A Clausal Resolution Method for CTL Branching Time Temporal Logic. *Journal of Experimental and Theoretical Artificial Intelligence.*, 11:77–93, 1999.

[5] J. Bradfield. and C. Stirling. Modal logics and mu-calculi. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 293–330. Elsevier, North-Holland, 2001.

[6] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics.*, pages 996–1072. Elsevier, 1990.

[7] E. A. Emerson. Automated reasoning about reactive systems. In *Logics for Concurrency: Structures Versus Automata, Proc. of International Workshop*, volume 1043 of Lecture Notes in Computer Science, pages 41–101. Springer, 1996.

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

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

[10] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. In *Proceedings of STOC 1984*, pages 14–24, 1984.

[11] M. Fisher. A Resolution Method for Temporal Logic. In *Proc. of the XII International Joint Conference on Artificial Intelligence (IJCAI)*, pages 99–104, 1991.

[12] P. Wolper. On the relation of programs and computations to models of temporal logic. In L. Bolc and A. Szałas, editors, *Time and Logic, a computational approach*, chapter 3, pages 131–178. UCL Press Limited, 1995.