Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Applied Logic 4 (2006) 141–167

JOURNAL OF APPLIED LOGIC

www.elsevier.com/locate/jal

A clausal resolution method for extended computation tree logic ECTL

Alexander Bolotov [*], Artie Basukoski

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

Available online 21 July 2005

Abstract

A temporal clausal resolution method was originally developed for linear time temporal logic and further extended to the branching-time framework of Computation Tree Logic (CTL). In this paper, following our general idea to expand the applicability of this efficient method to more expressive formalisms useful in a variety of applications in computer science and AI requiring branching time logics, we define a clausal resolution technique for Extended Computation Tree Logic (ECTL). The branching-time temporal logic ECTL is strictly more expressive than CTL in allowing fairness operators. The key elements of the resolution method for ECTL, namely the clausal normal form, the concepts of step resolution and a temporal resolution, are introduced and justified with respect to this new framework. Although in developing these components we incorporate many of the techniques defined for CTL, we need novel mechanisms in order to capture fairness together with the limit closure property of the underlying tree models. We accompany our presentation of the relevant techniques by examples of the application of the temporal resolution method. Finally, we provide a correctness argument and consider future work discussing an extension of the method yet further, to the logic CTL*, the most powerful logic of this class.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Branching-time; Automated deduction; Resolution; Program specification and verification

[*] Corresponding author.
E-mail addresses: a.bolotov@wmin.ac.uk (A. Bolotov), a.basukoski@wmin.ac.uk (A. Basukoski).

1570-8683/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.jal.2005.06.003

1. Introduction

A Computation Tree Logic (CTL), first proposed in [9], and its extensions have shown to play a significant role in potential applications [11]. CTL does not permit boolean combinations of formulae with temporal operators or their nesting. Two combinations of future time temporal operators $\Diamond$ ('sometime') and $\Box$ ('always'), are useful in expressing *fairness* [10]: $\Diamond\Box p$ (p is true along the path of the computation except possibly some finite initial interval of it) and $\Box\Diamond p$ (p is true along the computation path at infinitely many moments of time).

The logic ECTL (Extended CTL [12]) bridges this gap in CTL expressiveness, admitting simple fairness constraints. While ECTL is strictly more expressive than CTL, their syntactic and semantic features have much in common.

In [5,6] a clausal resolution approach to CTL was developed, extending the original definition of the method for the linear-time case [14]. In this paper, following our general aim to expand the applicability of the method to more expressive formalisms, we define it for the logic ECTL.[1] As a normal form for ECTL, called $\mathrm{SNF_{CTL}}$, we utilise the Separated Normal Form developed for CTL formulae. This enables us to apply the resolution technique defined over $\mathrm{SNF_{CTL}}$ as the refutation technique for ECTL formulae.

The main contribution of this paper is the extension of the set of rules used to translate CTL formulae into $\mathrm{SNF_{CTL}}$ by *a novel* transformation technique to cope with ECTL fairness. $\mathrm{SNF_{CTL}}$ can be used for more expressive formalisms, such as ECTL: in translating CTL or ECTL formulae into our normal form, similarly to the linear time case [7], we derive propositional formulae that are existentially quantified, and to utilise the normal form as part of a proof, we effectively skolemize them producing temporal formulae without any quantification.

The structure of the paper is as follows. In Section 2 we introduce the logic ECTL, outlining the ECTL syntax in Section 2.1, its semantics in Section 2.2 and those properties of ECTL syntax and semantics that are important for our analysis in Section 2.3. Further, in Section 3, we review $\mathrm{SNF_{CTL}}$. In the next section, Section 4 we describe the translation of ECTL formulae into $\mathrm{SNF_{CTL}}$. The translation algorithm which includes a novel transformation technique to cope with ECTL fairness constraints is given in Section 4.1. Main rules invoked in this algorithm are given in Section 4.2. Some of these rules are used in the example transformation which can be found in Section 4.3. We conclude this section providing in Section 4.4 the correctness argument. Note also that in this paper we present the full correctness argument which bridges the gap contained in the correctness proof contained in [3], where we only show that the transformation procedure preserves satisfiability: now we also establish that it preserves unsatisfiability. Having provided the translation of ECTL formulae into $\mathrm{SNF_{CTL}}$, we represent all temporal statements within ECTL as sets of $\mathrm{SNF_{CTL}}$ clauses. Now, in order to achieve a refutation, we incorporate the temporal resolution method already defined over $\mathrm{SNF_{CTL}}$ in [2,6]. The method is outlined

[1] Note that while working on the preparation of this article the authors have developed the resolution technique for the logic ECTL^{+}, the extension of ECTL, allowing Boolean combination of fairness constraints [4], and are currently working on the extended version of this paper.

in Section 5, where we also apply the temporal resolution to a set of $\mathrm{SNF_{CTL}}$ clauses (previously obtained in Section 4.3). Finally, in Section 6, we draw conclusions and discuss future work.

2. Syntax and semantics of ECTL

2.1. ECTL syntax

For clarity we will introduce the normal form for ECTL (see Section 3) and the resolution rules defined over a set of clauses in normal form (see Section 5), based on the set of classical logic operators $\wedge$, $\vee$, $\Rightarrow$, $\neg$, future time temporal operators $\Box$ (always), $\Diamond$ (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). Thus, to unify the presentation, here we define the language of ECTL also based upon this extended set of operators.

We fix a countable set, *Prop*, of atomic propositions. In the syntax of ECTL we distinguish *state* (S) and *path* (P) formulae, such that well formed formulae are state formulae. These classes of formulae are inductively defined below (where C is a formula of classical propositional logic)

$$S ::= C|S \wedge S|S \vee S|S \Rightarrow S|\neg S|\mathbf{A}P|\mathbf{E}P$$
$$P ::= \Box S|\Diamond S|\bigcirc S|S\mathcal{U}S|S\mathcal{W}S|\Box\Diamond S|\Diamond\Box S$$

Thus, ECTL has a richer syntax than CTL allowing nesting of $\Box$ and $\Diamond$ in the scope of path quantifiers. Examples of ECTL formulae are $\mathbf{A}\Diamond\Box B$, $\mathbf{A}\Box\Diamond B$, $\mathbf{E}\Diamond\Box B$ and $\mathbf{E}\Box\Diamond B$ (where B is any ECTL formula), which express the fairness properties.

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 [10].

2.2. ECTL semantics

Let us introduce the notation of tree structures, the underlying structures of time assumed for branching-time logics, which we utilise in our presentation.

Definition 1. A *tree* 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 $\mathcal{M}$. 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 $\mathit{Suf}(\chi_{s_i}, s_j)$.

Definition 2 *(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.

In a general case a state of a tree can have an infinite number of successors. However, following [10, p. 1011], trees with arbitrary, even uncountable, branching, "as far as our branching temporal logic are concerned, are indistinguishable from trees with finite, even bounded, branching". Thus, without loss of generality, we assume that underlying ECTL tree models are of at most countable branching.

Definition 3 *(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.

We interpret a well-formed ECTL formula in a structure $\mathcal{M} = \langle S, R, s_0, X, L\rangle$, where (S, R) is a 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 [10], 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 $\chi_j = s_0, s_1, s_2, \ldots, \chi_j \in X$ if, and only if, for every i, $R(s_i, s_{i+1})$;
- a tree (S, R) is of at most countable branching.

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

$\langle\mathcal{M}, s_i\rangle$	$\models p$	iff	$p \in L(s_i)$, for $p \in \mathit{Prop}$.
$\langle\mathcal{M}, s_i\rangle$	$\models \neg A$	iff	$\langle\mathcal{M}, s_i\rangle \not\models A$
$\langle\mathcal{M}, s_i\rangle$	$\models A \wedge B$	iff	$\langle\mathcal{M}, s_i\rangle \models A$ and $\langle\mathcal{M}, s_i\rangle \models B$
$\langle\mathcal{M}, s_i\rangle$	$\models A \vee B$	iff	$\langle\mathcal{M}, s_i\rangle \models A$ or $\langle\mathcal{M}, s_i\rangle \models B$
$\langle\mathcal{M}, s_i\rangle$	$\models A \Rightarrow B$	iff	$\langle\mathcal{M}, s_i\rangle \not\models A$ or $\langle\mathcal{M}, s_i\rangle \models B$
$\langle\mathcal{M}, s_i\rangle$	$\models \mathbf{A}B$	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	there exists χ_{s_i} such that $\langle\mathcal{M}, \chi_{s_i}\rangle \models B$
$\langle\mathcal{M}, \chi_{s_i}\rangle$	$\models A$	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 $i \leqslant j$ then $\langle\mathcal{M}, \mathit{Suf}(\chi_{s_i}, s_j)\rangle \models B$.
$\langle\mathcal{M}, \chi_{s_i}\rangle$	$\models \Diamond B$	iff	there exists $s_j \in \chi_{s_i}$ such that $i \leqslant j$ and $\langle\mathcal{M}, \mathit{Suf}(\chi_{s_i}, s_j)\rangle \models B$.
$\langle\mathcal{M}, \chi_{s_i}\rangle$	$\models \bigcirc B$	iff	$\langle\mathcal{M}, \mathit{Suf}(\chi_{s_i}, s_{i+1})\rangle \models B$.
$\langle\mathcal{M}, \chi_{s_i}\rangle$	$\models A\mathcal{U}B$	iff	there exists $s_j \in \chi_{s_i}$ such that $i \leqslant j$ and $\langle\mathcal{M}, \mathit{Suf}(\chi_{s_i}, s_j)\rangle \models B$ and for each $s_k \in \chi_{s_i}$, if $i \leqslant k < j$ then $\langle\mathcal{M}, \mathit{Suf}(\chi_{s_i}, s_k)\rangle \models A$.
$\langle\mathcal{M}, \chi_{s_i}\rangle$	$\models A\mathcal{W}B$	iff	$\langle\mathcal{M}, \chi_{s_i}\rangle \models \Box A$ or $\langle\mathcal{M}, \chi_{s_i}\rangle \models A\mathcal{U}B$

Fig. 1. ECTL semantics.

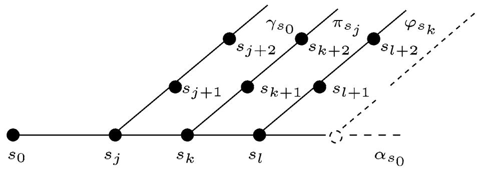

Fig. 2. Limit closure.

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.

As an example, let us consider the following satisfiable ECTL formula

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

A model, $\mathcal{M}$, for this formula (see Fig. 2) can be derived as follows. Let for the states along α_{s_0}, the following holds: $k = j+1, l = k+1, \ldots$; let $\Box p$ be satisfied at $\mathit{Suf}(\alpha_{s_0}, s_j)$ and also at $\mathit{Suf}(\gamma_{s_0}, s_{j+2}), \mathit{Suf}(\pi_{s_j}, s_{k+2}), \mathit{Suf}(\phi_{s_k}, s_{l+2}), \ldots$. Finally, let $s_{j+1}, s_{k+1}, s_{l+1}, \ldots$ along paths $\gamma, \pi, \phi, \ldots$, respectively, satisfy $\neg p$.

Note that if we change the first conjunct of formula (1) to $\mathbf{A}\Diamond\mathbf{A}\Box p$ then the whole formula becomes unsatisfiable.

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 [10], 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” [10]. This corresponds to limit closure property, meaning that for any fullpath γ_{s_0} and any paths $\pi_{s_j}, \phi_{s_k}, \ldots$ 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 Fig. 2): there exists an infinite path α_{s_0} that is a limit of the prefixes $[s_0, s_j], [s_j, s_k], [s_k, s_l], \ldots$.

In our definition of an ECTL model structure M the set of fullpaths X is R-generable. Therefore, following [10], it satisfies all three closure properties, i.e., it is suffix, fusion and limit closed.

2.3. Some useful features of ECTL

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

Fairness constraints. Validity of the following equivalences can be easily shown in ECTL semantics:

$$\mathbf{A}\Box\Diamond B \equiv \mathbf{A}\Box\mathbf{A}\Diamond B, \qquad \mathbf{E}\Diamond\Box B \equiv \mathbf{E}\Diamond\mathbf{E}\Box B \tag{2}$$

Therefore, $\mathbf{A}\Box\Diamond B$ and $\mathbf{E}\Diamond\Box B$ have their CTL counterparts. However, $\mathbf{E}\Box\Diamond B$ and $\mathbf{A}\Diamond\Box B$ have no analogues in CTL [10]. Note that in the case of $\mathbf{E}\Box\Diamond$, the $\Diamond$ operator is in the scope of the $\Box$ operator, which is a maximal fixpoint prefixed by the '**E**' quantifier. In the second case, the $\Box$ operator is in the scope of the $\Diamond$ operator, which is a minimal fixpoint and is prefixed by the '**A**' quantifier. These nestings of temporal operators significantly affect the renaming of the embedded paths subformulae in the corresponding ECTL fairness constraints.

Notation.

- 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 expression of the type $\mathbf{PT}$ or $\mathbf{PT}^2$ is called *a basic* CTL *modality*. A class of *basic* ECTL *modalities* consists of basic CTL modalities, enriched by ECTL fairness constraints, $\mathbf{P}\Box\Diamond$ and $\mathbf{P}\Diamond\Box$.
- Let F be an ECTL formula and let F_i be its subformula with a path quantifier as its main operator. We will abbreviate the latter by **P**-*embedded subformula of* F.
- A *literal* is an atomic proposition or its negations.
- We will use the symbol $=$ in the expression $A = B$ to refer to the graphical equivalence of formulae A and B, while $A \equiv B$ would mean the logical equivalence, i.e., it abbreviates $(A \Rightarrow B) \wedge (B \Rightarrow A)$.

As we will see in Section 3, the idea behind the normal form for ECTL is to identify the core operators, $\mathbf{P}\bigcirc$ and $\mathbf{P}\Diamond$, to enable us to generate formulae relevant to either the first state in a model, or to all subsequent states in a model. Therefore, an important part of the transformation procedure for ECTL formulae into the normal form is simplifying the structures of the given ECTL formulae. Some of these transformations deal with embedded state subformulae of ECTL formulae.

Managing embedded state subformulae. 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 [13] showed that any CTL* formula F can be transformed into F' such that $N(F') = 2$ and F is satisfiable if, and only if, F' is satisfiable. This can be achieved by a continuous renaming of the **P**-embedded state subformulae. The result is obviously valid for the logic ECTL, and below we introduce a corresponding recursive procedure *Red*.

Definition 7 *(Reduction of the path quantifier nesting).* Given an ECTL formula F such that $N(F) > 2$, the following procedure reduces the nesting of path quantifiers in F to the degree 2: $\mathit{Red}[F] = \mathbf{A}\Box(x_1 \equiv S_1) \wedge \mathit{Red}[F(S_1/x_1)]$, where S_1 is the designated **P**-embedded state subformula of F, x_1 is a new atomic proposition and $F(S_1/x_1)$ is a result of the replacement of S_1 in F by x_1. If $N(S_i) = 1$ then the procedure terminates.

For example, given $F = \mathbf{A}\Diamond(\mathbf{E}\Box\Diamond\neg p \wedge \mathbf{A}\Diamond\mathbf{A}\Box p)$ we can obtain $\mathit{Red}[F]$ as follows

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

Therefore, procedure *Red* terminates here producing the conjunction of formulae such that each of them has a degree of nesting of path quantifiers at most 2.

The following Proposition 1 is due to Emerson and Sistla [13].

Proposition 1 *(Correctness of the Reduction procedure). For any ECTL formula F, F is satisfiable if, and only if, $\mathit{Red}(F)$ is satisfiable, where Red is introduced in Definition 7.*

Since normal form for ECTL is invoked as part of the resolution method, and similarly to classical resolution, the resolution based refutation commences here with the negation of a given ECTL formula (see Section 5), we will aim at translating this negation into negation normal form.

Definition 8 *(Negation normal form for ECTL).* An ECTL formula is in the negation normal form if every negation operation applies to an atomic proposition.

Using the standard technique we can translate an ECTL formula F into its negation normal form, $\mathrm{NNF}_{\mathrm{ECTL}}(F)$ [10] preserving both satisfiability and unsatisfiability.

Proposition 2 *(Correctness of $\mathrm{NNF}_{\mathrm{ECTL}}$). For any* ECTL *formula, F, F is satisfiable if, and only if,* $\mathrm{NNF}_{\mathrm{ECTL}}(F)$ *is satisfiable.*

Fixpoint characterization of basic CTL modalities. Our translation to $\mathrm{SNF}_{\mathrm{CTL}}$ and temporal resolution rules are essentially based upon the fixpoint characterizations of basic

CTL modalities (see [8]). The corresponding definitions are given below, where maximal fixpoint operator is abbreviated by "ν" and minimal fixpoint operator by "μ":

$$\begin{aligned}
&\mathbf{E}\Box p = \nu\zeta(p \wedge \mathbf{E}\bigcirc\zeta)\\
&\mathbf{A}\Box p = \nu\eta(p \wedge \mathbf{A}\bigcirc\eta)\\
&\mathbf{E}(p\mathcal{W}q) = \nu\kappa\big(q \vee (p \wedge \mathbf{E}\bigcirc\kappa)\big)\\
&\mathbf{A}(p\mathcal{W}q) = \nu\xi\big(q \vee (p \wedge \mathbf{A}\bigcirc\xi)\big)
\end{aligned} \tag{3}$$

$$\begin{aligned}
&\mathbf{E}\Diamond p = \mu\rho(p \vee \mathbf{E}\bigcirc\rho)\\
&\mathbf{A}\Diamond p = \mu\tau(p \vee \mathbf{A}\bigcirc\tau)\\
&\mathbf{E}(p\mathcal{U}q) = \mu\chi\big(q \vee (p \wedge \mathbf{E}\bigcirc\chi)\big)\\
&\mathbf{A}(p\mathcal{U}q) = \mu\delta\big(q \vee (p \wedge \mathbf{A}\bigcirc\delta)\big)
\end{aligned} \tag{4}$$

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*.

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 we introduced the notion of satisfiability and validity of ECTL formulae in relation to $\langle\mathcal{M}, s_0\rangle$. Now, let us, following [15], 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 [15] 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 $\{(s, si) \mid s \in [k]^\star, i \in \mathcal{K}\}$, 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 $R = \{(s, si) \mid s \in [k]^\star, i \in \mathcal{K}\}$. 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 [15], 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 3 given below collects the results of [15, Lemma 3.5, p. 145].

Proposition 3 *(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 $\langle \mathcal{M}', \lambda\rangle$. We will essentially use these results for the formulation of the transformation rule managing ECTL fairness constraints, namely, formulae that contain $\mathbf{A}\Diamond\Box$—see Section 4.2. The results are also central to the correctness proof of this transformation presented in Section 4.4.

3. Normal form for ECTL

As a normal form for ECTL we utilise a clausal normal form, defined for the logic CTL, $\mathrm{SNF}_{\mathrm{CTL}}$, which has been developed in [2,6]. The idea behind $\mathrm{SNF}_{\mathrm{CTL}}$ is to identify the core operators, $\mathbf{P}\bigcirc$ and $\mathbf{P}\Diamond$, which enables us to generate formulae relevant to either the first state in a model, or to all subsequent states in a model. Therefore, as an important part of the transformation procedure for ECTL formulae into $\mathrm{SNF}_{\mathrm{CTL}}$ we incorporate removal of all other, unwanted modalities $\mathbf{A}\Box$, $\mathbf{E}\Box$, $\mathbf{A}\mathcal{U}$, $\mathbf{E}\mathcal{U}$, $\mathbf{A}\mathcal{W}$, $\mathbf{E}\mathcal{W}$ (see Section 4.2).

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 $\mathrm{SNF}_{\mathrm{CTL}}$ are indices of the type $\langle \mathsf{LC}(\mathsf{ind})\rangle$ which represent a limit closure of $\langle \mathsf{ind}\rangle$. All formulae of $\mathrm{SNF}_{\mathrm{CTL}}$ 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 is related to the branching factor of the canonical model for the clauses and will be explained later.

The $\mathrm{SNF}_{\mathrm{CTL}}$ 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 $\mathit{SNF}_{\mathit{CTL}}$*). $\mathrm{SNF}_{\mathrm{CTL}}$ is a set of formulae

$$\mathbf{A}\Box\Big[\bigwedge_i (P_i \Rightarrow F_i)\Big]$$

where each of the *clauses* $P_i \Rightarrow F_i$ is further restricted as below, each α_i, β_j or l is a literal, **true** or **false** and $\langle \mathsf{ind}\rangle \in \mathsf{IND}$ is some index.

$$\mathbf{start} \Rightarrow \bigvee_{j=1}^{k} \beta_j \qquad \text{an Initial Clause}$$

Step Clauses

$$\bigwedge_{i=1}^{m} \alpha_i \Rightarrow \mathbf{A}\bigcirc\left[\bigvee_{j=1}^{n} \beta_j\right] \qquad \text{an } \mathbf{A} \text{ step clause}$$

$$\bigwedge_{i=1}^{m} \alpha_i \Rightarrow \mathbf{E}\bigcirc\left[\bigvee_{j=1}^{n} \beta_j\right]_{\langle \mathsf{ind} \rangle} \qquad \text{an } \mathbf{E} \text{ step clause}$$

Sometime Clauses

$$\bigwedge_{i=1}^{m} \alpha_i \Rightarrow \mathbf{A}\Diamond l \qquad \text{an } \mathbf{A} \text{ sometime clause}$$

$$\bigwedge_{i=1}^{m} \alpha_i \Rightarrow \mathbf{E}\Diamond l_{\langle \mathsf{LC(ind)} \rangle} \qquad \text{an } \mathbf{E} \text{ sometime clause}$$

Interpreting SNF$_{CTL}$. An initial SNF$_{\mathrm{CTL}}$ clause, $\mathbf{start} \Rightarrow F$, is understood as "$F$ is satisfied at the initial state of some model $\mathcal{M}$". Any other SNF$_{\mathrm{CTL}}$ clause is interpreted taking also into account that it occurs in the scope of $\mathbf{A}\Box$.

Thus, a clause $\mathbf{A}\Box(x \Rightarrow \mathbf{A}\bigcirc p)$ (a model for which is given in Fig. 3, Diagram 1) 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*".

Recall that following Proposition 3, for a set, R, of clauses with n existential path quantifiers, we have an $(n+1)$-ary canonical model for R. Now, associating every $k \in 1 \ldots n$ with a unique index $ind_k \in IND$, we label each $\mathbf{E}$ step clause with the unique ind_k and each $\mathbf{E}$ sometime clause with the unique $LC(ind_k)$. Thus, a clause $\mathbf{A}\Box(x \Rightarrow \mathbf{E}\bigcirc q_{\langle \mathsf{ind} \rangle})$ (see Fig. 3, Diagram 1) is understood 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 some path $\langle \mathsf{ind} \rangle$ which departs from s_i*".

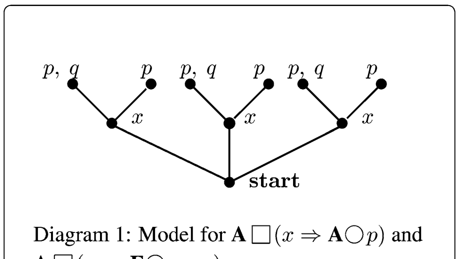

Diagram 1: Model for $\mathbf{A}\Box(x \Rightarrow \mathbf{A}\bigcirc p)$ and $\mathbf{A}\Box(x \Rightarrow \mathbf{E}\bigcirc q_{\langle \mathsf{ind} \rangle})$

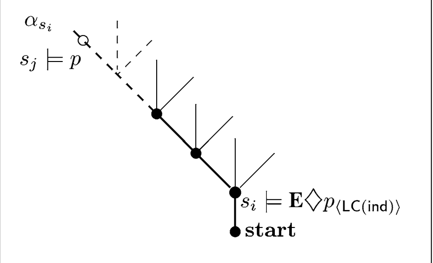
α_{s_i}
$s_j \models p$
$s_i \models \mathbf{E}\Diamond p_{\langle \mathsf{LC(ind)} \rangle}$
start
Diagram 2: SNF$_{\mathrm{CTL}}$ $\mathbf{E}\Diamond$ Clauses

Fig. 3. Interpretation of step and sometime clauses.

Finally, $\mathbf{A}\Box(x \Rightarrow \mathbf{E}\Diamond p_{\langle \mathsf{LC(ind)} \rangle})$ (see Fig. 3, Diagram 2) 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*".

4. Transformation of ECTL formulae into SNF$_{\mathrm{CTL}}$

As we have already mentioned, an important part of the transformation procedure for ECTL formulae includes removal of all unwanted modalities $\mathbf{A}\Box$, $\mathbf{E}\Box \ldots$. The corresponding rules are formulated in such way that these basic CTL modalities are removed being applied to literals. Thus, a significant part of the translation aims at simplifying the structure of formulae preparing them for application of removal rules. While many of these methods were developed in our previous work [2,6], we here introduce a novel technique to cope with ECTL fairness constraints.

4.1. Description of the transformation algorithm

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 $C = \mathrm{NNF}_{\mathrm{ECTL}}(\neg G)$. Now we introduce the transformation procedure

$$\tau = [\tau_2[\tau_1[C]]]$$

to be applied to C, where τ_1 and τ_2 are described respectively by the steps 1–2 and 3–7 below.

1. Anchor C to **start** and apply the initial renaming rule obtaining $\mathbf{A}\Box((\mathbf{start} \Rightarrow x_0) \wedge (x_0 \Rightarrow C))$, where x_0 is a new atomic proposition.
2. Apply Eqs. (2) and then procedure *Red* (see Definition 7) to C. Thus, we derive a set of constraints of the following structure

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

   where P_j is a proposition, Q_j is either a purely classical formula or if Q_j contains an ECTL modality 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, or conjunction of literals, or **start**, Q_j is either a purely classical formula or $Q_j = \mathbf{PT}C_j$ or $Q_j = \mathbf{P}\Box\Diamond C_j$ or $Q_j = \mathbf{P}\Diamond\Box C_j$ or $Q_j = \mathbf{P}(C_{j_1}\mathbf{T}^2 C_{j_2})$, and C_j, C_{j_1} and C_{j_2} are purely classical formulae.
3. For every pre-clause $P_j \Rightarrow Q_j$, 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 $Q_j = \mathbf{P}\Box\Diamond C_j$ or $Q_j = \mathbf{P}\Diamond\Box C_j$ 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.
This can be achieved by continuous renaming of the embedded classical subformulae by auxiliary propositions together with some classical transformations.

4. Label each pre-clause containing the $\mathbf{E}\bigcirc$ modality by a unique index $\langle \mathsf{ind_i} \rangle \in \mathsf{IND}$ and any other pre-clause containing the $\mathbf{E}$ quantifier by a unique index $\langle \mathsf{LC(ind_j)} \rangle \in \mathsf{IND}$. Let *LIST_IND* be a list of all indices introduced during this labelling.
5. Transform pre-clauses containing $\mathbf{E}\Box\Diamond$ and $\mathbf{A}\Diamond\Box$.
6. Remove all unwanted basic CTL modalities.
Steps 5 and 6 are described in the next section.
7. Derive the desired form of $\mathrm{SNF_{CTL}}$ 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.
– 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 $\mathit{SNF_{CTL}}$

In the transformation procedure τ outlined above, the first stage, the procedure τ_1, except for the application of equations (2) at step 2, is taken from the translation of CTL formulae to $\mathrm{SNF_{CTL}}$ [2]. In the procedure τ_2 we introduce novel techniques to cope with ECTL fairness constraints that do not have their CTL counterparts. Here we describe these techniques and recall some of those rules that will be used in our example given in Section 4.3. For the full set of rules preserved from the CTL the reader is referred to [2,6].

In the presentation below we omit the outer '$\mathbf{A}\Box$' connective that surrounds the conjunction of pre-clauses (note that any pre-clause is also a clause) 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 4 of the transformation procedure, we introduce labelling of the $\mathrm{SNF_{CTL}}$ pre-clauses containing the $\mathbf{E}$ quantifier: here we first label every pre-clause $P \Rightarrow \mathbf{E}\bigcirc Q$ by a unique index $\langle \mathsf{ind_i} \rangle$, indicating a 'direction' in which Q is satisfied, given that P is satisfied. Secondly, with any other pre-clause containing the $\mathbf{E}$ quantifier we associate a unique index $\langle \mathsf{LC(ind_j)} \rangle$. The justification of the latter labelling is based upon fixpoint characterization of basic CTL modalities $\mathbf{E}\Box$, $\mathbf{E}\mathcal{W}$ and $\mathbf{E}\mathcal{U}$ (see Eqs. (3) and (4)).

Assume that a pre-clause $P \Rightarrow \mathbf{E}\Box y$ has been derived at some stage of the transformation procedure. Since $\mathbf{E}\Box y$ is a maximal fixpoint of the equation $\nu\zeta(y \wedge \mathbf{E}\bigcirc\zeta)$, we can represent this recursion by the following set of constraints:

$$\begin{aligned} &P \Rightarrow y \wedge x \\ &x \Rightarrow \mathbf{E}\bigcirc(y \wedge x)_{\langle \mathsf{ind} \rangle} \end{aligned} \tag{5}$$

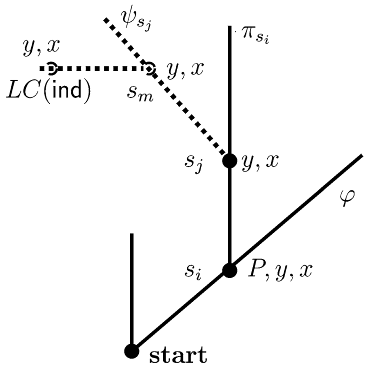

Fig. 4. Labelling ECTL formulae: the *LC* index.

where we introduce a new atomic proposition, x, and require that the conjunction $y \wedge x$ also occurs at those moments where P itself is satisfied. The second constraint, $x \Rightarrow \mathbf{E}\bigcirc(y \wedge x)$, represents a loop in y, i.e., the situation, where y occurs from some point at all subsequent states along some path in the model (given that x is satisfied at that point).

Now, labelling $x \Rightarrow \mathbf{E}\bigcirc(y \wedge x)$ by a new index, $\langle\mathsf{ind}\rangle$, and noting that pre-clauses are in the scope of the outer $\mathbf{A}\Box$, we can show that $P \Rightarrow \mathbf{E}\Box y$ is satisfiable in some model, $\mathcal{M}$, if, and only if, there is a model $\mathcal{M}'$ which satisfies both formulae in (5). Here we present a proof establishing that if $P \Rightarrow \mathbf{E}\Box y$ is satisfiable in a model $\mathcal{M}$ then there is a model $\mathcal{M}'$ which satisfies both formulae in (5).

The satisfiability of pre-clause $P \Rightarrow \mathbf{E}\Box y$ in a model $\mathcal{M}$ would mean

'for any fullpath χ and any state $s_k \in \chi$ $(0 \leqslant k)$, if $\langle\mathcal{M}, s_k\rangle \models P$ then $\langle\mathcal{M}, s_k\rangle \models \mathbf{E}\Box y$'

Choose arbitrarily a fullpath φ (see Fig. 4).

If P is never satisfied along φ then let $\mathcal{M}'$ be the same as $\mathcal{M}$ except for a new proposition x such that x is false everywhere along φ. Thus, we obtain

$$\langle\mathcal{M}', \chi\rangle \models \Box\big(P \Rightarrow (y \wedge x)\big),$$
$$\langle\mathcal{M}', \chi\rangle \models \Box\big(x \Rightarrow \mathbf{E}\bigcirc(y \wedge x)_{\langle\mathsf{ind}\rangle}\big)$$

regardless of the indices since the left hand side of each implication is false. Alternatively, let $s_i \in \varphi$ be the first moment along φ satisfying P. In this case there must be a path π_{s_i} (associated with $\langle\mathsf{ind}\rangle$) such that $\langle\mathcal{M}, \pi_{s_i}\rangle \models \Box y$. Due to the fusion closure property, there is a fullpath $[s_0, s_i] \circ \pi_{s_i}$, where '$\circ$' is a concatenation of $[s_0, s_i]$ and π_{s_i}. Now we define a model $\mathcal{M}'$ to be the same as $\mathcal{M}$ except for a new proposition x such that for any state $s_n \in [s_0, s_i] \circ \pi_{s_i}$, if $i \leqslant n$ then $\langle\mathcal{M}', s_n\rangle \models x$ else $\langle\mathcal{M}', s_n\rangle \not\models x$. Now we derive that s_j, the successor of s_i on path π_{s_i}, satisfies $y \wedge x$. Thus, setting in the conditions for $P \Rightarrow \mathbf{E}\bigcirc y$ that $\chi = [s_0, s_i] \circ \pi_{s_i}$ and $k = j$, we conclude that $\langle\mathcal{M}, s_j\rangle \models \mathbf{E}\bigcirc(y \wedge x)_{\langle\mathsf{ind}\rangle}$. Therefore, there is a path ψ_{s_j} associated with $\langle\mathsf{ind}\rangle$ such that there is a state, next to s_i, say s_m, on this path, which satisfies $y \wedge x$. Continuing to reason in this way, according to the limit closure property, we must have in the model a path, $\langle LC(\mathsf{ind})\rangle$, going through the states $s_i, s_j, s_m \ldots$. Each state along $\langle LC(\mathsf{ind})\rangle$ satisfies $y \wedge x$. Therefore, we have identified a path which satisfies $\mathbf{E}\Box y$, which enables us to label pre-clause $P \Rightarrow \mathbf{E}\Box y$ by $\langle LC(\mathsf{ind})\rangle$. Note also that this justifies that $\Box(x \Rightarrow \mathbf{E}\bigcirc(y \wedge x)_{\mathsf{ind}})$ indeed represents a loop in y on the path $\langle LC(\mathsf{ind})\rangle$. Searching for loops is essential for application of resolution rules, see Section 5.

Providing analogous reasoning, we can justify the labelling of pre-clauses containing $\mathbf{E}\mathcal{W}$, taking into account their definitions as maximal fixpoints, and the labelling of pre-clauses containing $\mathbf{E}\Diamond$ and $\mathbf{E}\mathcal{U}$ modalities based upon their definitions as minimal fixpoints.

Obviously, this representations of basic CTL modalities as sets of pre-clauses allows us to formulate corresponding rules to substitute basic CTL modalities by their fixpoint definitions. Thus, given $P \Rightarrow \mathbf{E}\Box y_{\langle LC(\mathsf{ind})\rangle}$, we apply (5) to remove the $\mathbf{E}\Box$ modality as follows (in the formulation of the rule below x is a new atomic proposition):

Removal of $\mathbf{E}\Box$.

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

Additionally, we give the removal rule for $\mathbf{E}\mathcal{W}$ referring the reader to [2,6] for the formulation of the full set of rules to remove basic CTL modalities.

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

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

where x is a new atomic proposition.

Managing embedded path subformulae in ECTL. The rules to rename purely path formulae embedded in ECTL fairness constraints are based upon our analysis of the problematic variety of nesting of temporal operators in ECTL (see Section 2.3). Thus, when renaming $\Diamond P$ within $\mathbf{E}\Box\Diamond P$ or $\Box P$ within $\mathbf{A}\Diamond\Box P$ by a new variable x, we must be sure that x and $\Box P$ in the former case, and x and $\Diamond P$ in the latter case, occur along the same path. Second, we must establish a link between satisfiability of x and $\Diamond P$ ($\Box P$), i.e., any state in a model which satisfies x should also satisfy $\Diamond P$ ($\Box P$). These observations have led us to the following formulation of the renaming rules.

Renaming: the $\mathbf{E}\Box\Diamond$ *case*.

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

where x is a new atomic proposition.

Applying this rule, the label, $\langle LC(\mathsf{ind})\rangle$ introduced for the premise at stage 4 of the transformation procedure, is preserved for both components of the conclusion.

Things are much more difficult when we deal with the $\mathbf{A}\Diamond\Box$ constraint. Recall that once we have provided the labelling of formulae at stage 4 of the transformation procedure, the number of indices is equal to the number of different **E** pre-clauses. Now we use this information about the number of existential path quantifiers based upon proof of Proposition 3, namely, from the fact that "one needs only sufficient paths from each state of a model to satisfy all the existential path formulae that have to be true in that state. Moreover the number of existential state formulae that can appear in a formula is bounded by the number of path quantifiers in that formula" [15].

Renaming: the $\mathbf{A}\Diamond\Box$ *case.* Let $LIST\_IND = \langle \mathsf{ind_1}\rangle, \ldots, \langle \mathsf{ind_n}\rangle$. 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 $\langle LC(\mathsf{ind})\rangle$ (which can be easily justified).

Now, based on Proposition 3, we rename the $\Box Q$ subformula of $\mathbf{A}\Diamond\Box Q$ as follows (if the number of indices in $LIST\_IND$, $n = 0$, then we create $LIST\_IND = \langle \mathsf{ind}\rangle$ with the new index $\langle \mathsf{ind}\rangle$).

if $n = 0$

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

if $n > 0$

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

where $x, x_1, \ldots, x_n$ are new atomic propositions.

Next we present another useful rule, called '*Temporising*', which allows us to introduce a temporal context, rewriting into $\mathrm{SNF_{CTL}}$ purely classical formulae of the type $Q \Rightarrow P$.

Temporising.

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

Finally, we utilize two rules allowing us to distribute the $\mathbf{A}\bigcirc$ and $\mathbf{E}\bigcirc$ modalities over conjunction. In the latter rule, which will be used in our example, we again, incorporate indices: given that the premise of this 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.

Distributing $\mathbf{A}\bigcirc$ *and* $\mathbf{E}\bigcirc$ *over conjunction.*

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

4.3. Example transformation

As an example we consider the application of the temporal resolution method, to check the validity of the following ECTL formula:

$$\mathbf{A}\Diamond\Box p \Rightarrow \mathbf{A}\Diamond p \tag{6}$$

To check that (6) is valid we negate it, obtaining

$$\neg(\mathbf{A}\Diamond\Box p \Rightarrow \mathbf{A}\Diamond p) \tag{7}$$

and then translate (7) into *Negation Normal Form*

$$\mathbf{A}\Diamond\Box p \wedge \mathbf{E}\Box\neg p \tag{8}$$

Following the translation algorithm, we obtain steps 0–2 below, where x is a new atomic proposition.

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

Now we split conjunction on the right hand side of the formula at step 2, generating steps 3–4.

3.	x	$\Rightarrow$	$\mathbf{A}\Diamond\Box p$	from 2, splitting $\wedge$
4.	x	$\Rightarrow$	$\mathbf{E}\Box\neg p$	from 2, splitting $\wedge$

At this stage we first label pre-clause 4 by a new label, $\langle LC(f)\rangle$ creating *LIST_ IND* and then rename $\Box p$ in 3, introducing a new variable, l.

5.	x	$\Rightarrow$	$\mathbf{E}\Diamond l_{\langle LC(\mathsf{f})\rangle}$	from 3, Renaming : $\mathbf{A}\Diamond\Box$ case
6.	l	$\Rightarrow$	$\mathbf{E}\Box p_{\langle LC(\mathsf{f})\rangle}$	from 3, Renaming : $\mathbf{A}\Diamond\Box$ case

Now we must first apply the $\mathbf{E}\Box$ removal rule to 4, introducing a new variable, y, thus, deriving steps 7 and 8 below.

7.	x	$\Rightarrow$	$\neg p \wedge y$	from 4, Removal of $\mathbf{E}\Box$
8.	y	$\Rightarrow$	$\mathbf{E}\bigcirc(\neg p \wedge y)_{\langle \mathsf{f}\rangle}$	from 4, Removal of $\mathbf{E}\Box$

Similarly we remove the $\mathbf{E}\Box$ modality from 6 deriving 9–10 below (and introducing a new variable, r).

9.	l	$\Rightarrow$	$p \wedge r$	from 6, Removal of $\mathbf{E}\Box$
10.	r	$\Rightarrow$	$\mathbf{E}\bigcirc(p \wedge r)_{\langle \mathsf{f}\rangle}$	from 6, Removal of $\mathbf{E}\Box$

Now we split conjunctions on the right hand side of formulae 7 and 9.

11.	x	$\Rightarrow$	$\neg p$	from 7, splitting $\wedge$
12.	x	$\Rightarrow$	y	from 7, splitting $\wedge$
13.	l	$\Rightarrow$	p	from 9, splitting $\wedge$
14.	l	$\Rightarrow$	r	from 9, splitting $\wedge$

As steps 11–14 are purely classical expressions we introduce a temporal context incorporating the rule *Temporising*, deriving the steps below:

15. $\mathbf{start} \Rightarrow \neg x \vee \neg p$ from 11, Temporising
16. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg x \vee \neg p)$ from 11, Temporising
17. $\mathbf{start} \Rightarrow \neg x \vee y$ from 12, Temporising
18. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg x \vee y)$ from 12, Temporising
19. $\mathbf{start} \Rightarrow \neg l \vee p$ from 13, Temporising
20. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg l \vee p)$ from 13, Temporising
21. $\mathbf{start} \Rightarrow \neg l \vee r$ from 14, Temporising
22. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg l \vee r)$ from 14, Temporising

Finally, we distribute the $\mathbf{E}\bigcirc$ operator over conjunction in steps 8 and 10, preserving the labelling:

23. $y \Rightarrow \mathbf{E}\bigcirc\neg p_{\langle \mathsf{f} \rangle}$ from 8, Distributing $\mathbf{E}\bigcirc$ over $\wedge$
24. $y \Rightarrow \mathbf{E}\bigcirc y_{\langle \mathsf{f} \rangle}$ from 8, Distributing $\mathbf{E}\bigcirc$ over $\wedge$
25. $r \Rightarrow \mathbf{E}\bigcirc p_{\langle \mathsf{f} \rangle}$ from 10, Distributing $\mathbf{E}\bigcirc$ over $\wedge$
26. $r \Rightarrow \mathbf{E}\bigcirc r_{\langle \mathsf{f} \rangle}$ from 10, Distributing $\mathbf{E}\bigcirc$ over $\wedge$

The normal form of the given ECTL formula is represented by clauses 1, 5, 15–26.

4.4. *Correctness of the transformation of ECTL formulae into SNF$_{CTL}$*

Here we give the correctness proof for the transformation procedure $\tau = \tau_2(\tau_1(G))$ applied to an ECTL formula G. We first show that an ECTL formula G is satisfiable, if and only if, $\tau_1(G)$ is satisfiable (Lemma 1). Next, we will establish that the transformation procedure τ_2 preserves satisfiability (Lemma 2), and, finally, we prove that the converse of Lemma 2 is true, i.e., that given $\tau_2(\tau_1(G))$ is satisfiable so is $\tau_1(G)$ (Lemma 3).

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

Proof. Recall that procedure τ_1 consists of the following steps: anchoring the initial formula, G, to **start**, application of equivalences (2) and Procedure *Red*. Here we first must establish that given a satisfiable formula G, we derive a satisfiable formula $\mathbf{A}\square((\mathbf{start} \Rightarrow x_0) \wedge (x_0 \Rightarrow G))$, where x_0 is a new atomic proposition. This can be shown by taking a model $\mathcal{M}$ for the former and obtaining from it a model $\mathcal{M}'$ which differs from $\mathcal{M}$ only in the evaluation of x_0 which is set to be true at the initial state and false elsewhere. On the other hand, given a model that satisfies $\mathbf{A}\square((\mathbf{start} \Rightarrow x_0) \wedge (x_0 \Rightarrow G))$, we also have $\mathbf{start} \Rightarrow G$ satisfiable in the same model. Secondly, from the semantics of ECTL, application of equivalences (2) also preserves satisfiability and unsatisfiability. Finally, due to Proposition 1, procedure *Red* preserves satisfiability and unsatisfiability. Therefore, an ECTL formula, G, is satisfiable if, and only if, $\tau_1(G)$ is satisfiable. □

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

Recall that

- a formula G in *pre-clause form* is of the form $P_j \Rightarrow Q_j$, where P_j is a literal or **start**, Q_j is either a purely classical formula or $Q_j = \mathbf{PT}C_j$ or $Q_j = \mathbf{P}\Box\Diamond C_j$ or $Q_j = \mathbf{P}\Diamond\Box C_j$ or $Q_j = \mathbf{P}(C_{j_1}\mathbf{T}^2 C_{j_2})$, and C_j, C_{j_1} and C_{j_2} are purely classical formulae.
- any $\mathrm{SNF}_{\mathrm{CTL}}$ clause is also a formula in a pre-clause form.

We must show that any step of the transformation procedure τ_2 preserves satisfiability.

Proposition 4. *Let $\mathcal{M}$ be a model such that*

$$\langle \mathcal{M}, s_0\rangle \models \left[\bigwedge_i \mathbf{A}\Box R_i\right] \wedge \mathbf{A}\Box S$$

where each R_i and S are in a pre-clause form.

Then there exists a model $\mathcal{M}'$ such that

$$\langle \mathcal{M}', s_0\rangle \models \left[\bigwedge_i \mathbf{A}\Box R_i\right] \wedge \mathbf{A}\Box S'$$

where each R_i is in a pre-clause form and S' is a result of one step of the transformation $\tau_2[S]$.

Since $S = (P \Rightarrow Q)$ is in a pre-clause form, we must consider the cases, corresponding to possible applications of $\tau_2[\mathbf{A}\Box S]$. These cases are given by the stages 3–7 of the transformation algorithm described in Section 4. Here we outline the proof for the cases which represent the core transformation technique of the paper, i.e., where $Q = \mathbf{E}\Box\Diamond B$ (Case 1) and $Q = \mathbf{A}\Diamond\Box B$ (Case 2), omitting other cases, as proof of Proposition 4 for them repeats stages of the corresponding proof for CTL [2].

Case 1. Here we apply τ_2 in the following way (y is a new atomic proposition).

$$\frac{\tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Box\Diamond B_{\langle LC(\mathsf{ind})\rangle})]}{\begin{array}{l}\tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Box y_{\langle LC(\mathsf{ind})\rangle})]\\ \tau_2[\mathbf{A}\Box(y \Rightarrow \mathbf{E}\Diamond B_{\langle LC(\mathsf{ind})\rangle})]\end{array}}$$

Let $\mathcal{M}$ be a model which satisfies the condition of Proposition 4 in this case:

$$\langle \mathcal{M}, s_0\rangle \models \left[\bigwedge_i \mathbf{A}\Box R_i\right] \wedge \mathbf{A}\Box\big(P \Rightarrow \mathbf{E}\Box\Diamond B_{\langle LC(\mathsf{ind})\rangle}\big)$$

We show that there exists a model $\mathcal{M}'$ such that the following holds:

(a) $\langle \mathcal{M}', s_0\rangle \models \left[\bigwedge_i \mathbf{A}\Box R_i\right]$

(b) $\langle \mathcal{M}', s_0\rangle \models \mathbf{A}\Box\big(P \Rightarrow \mathbf{E}\Box y_{\langle LC(\mathsf{ind})\rangle}\big)$

(c) $\langle \mathcal{M}', s_0\rangle \models \mathbf{A}\Box\big(y \Rightarrow \mathbf{E}\Diamond B_{\langle LC(\mathsf{ind})\rangle}\big)$

In the corresponding proof we obtain a model $\mathcal{M}'$ from $\mathcal{M}$ by letting a new atomic proposition y to be satisfied in the relevant places and then establishing the conditions (a)–(c) taking into account the interpretation of the **E** clauses labelled with the '*LC*' type indices. Choose arbitrarily a fullpath φ in $\mathcal{M}$. If P is never satisfied along φ then let $\mathcal{M}'$ be the same as $\mathcal{M}$ accept for a new atomic proposition y which is false everywhere along φ. Thus, (a)–(c) are satisfied in $\mathcal{M}'$ as the left hand side of every implication of (a)–(c) is false. Alternatively, let $s_i \in \varphi$ be the first state along φ which satisfies P. Thus, according to $\mathrm{SNF}_{\mathrm{CTL}}$ semantics, there must be a path π_{s_i} associated with the $\langle LC(\mathsf{ind})\rangle$ such that $\langle \mathcal{M}, \pi_{s_i}\rangle \models \Box\Diamond B$. According to fusion closure there also must be a path $[s_0, s_i] \circ \pi_{s_i}$. Now we define a model $\mathcal{M}'$ to be the same as $\mathcal{M}$ accept for a new atomic proposition y such that for a path $[s_0, s_i] \circ \pi_{s_i}$ associated with the $\langle LC(\mathsf{ind})\rangle$, for any state $s_n \in [s_0, s_i] \circ \pi_{s_i}$, if $i \leqslant n$ then $\langle \mathcal{M}', s_n\rangle \models x$ else $\langle \mathcal{M}', s_n\rangle \not\models x$. This guarantees that (a)–(c) are now satisfied in $\mathcal{M}'$.

Case 2. Here we apply τ_2 in the following way

$$\begin{array}{l}
\tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{A}\Diamond\Box Q)] \\
\hline
\tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_1)_{\langle LC(\mathsf{ind_1})\rangle}] \\
\tau_2[\mathbf{A}\Box(x_1 \Rightarrow \mathbf{E}\Box Q)_{\langle LC(\mathsf{ind_1})\rangle}] \\
\ldots \\
\tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_n)_{\langle LC(\mathsf{ind_n})\rangle}] \\
\tau_2[\mathbf{A}\Box(x_n \Rightarrow \mathbf{E}\Box Q)_{\langle LC(\mathsf{ind_n})\rangle}]
\end{array}$$

Let $\mathcal{M}$ be a model which satisfies the condition of Proposition 4 in this case. According to Proposition 3, there exists an $(n+1)$-ary canonical model $\mathcal{M}'$, which also satisfies these conditions. Let $LC(\mathsf{ind_1}), \ldots LC(\mathsf{ind_n})$ be labels which correspond to linear interpretations of $\mathcal{M}'$. Note that each of these linear interpretations must satisfy $P \Rightarrow \mathbf{A}\Diamond\Box Q$. Now we update $\mathcal{M}'$ to $\mathcal{M}''$ labelling the states of the paths corresponding to linear interpretations by x_1 similar to the labelling carried out in the Case 1 considered above. Thus, we guarantee that all formulae in the conclusion of Proposition 4 in Case 2 are satisfied in $\mathcal{M}''$. Note also that, once the labelling at stage 4 of the transformation procedure has been provided, no more new indices will appear in the proof.

Lemma 3. *Given an ECTL formula G, if $\tau_2(\tau_1(G))$ is satisfiable then so is $\tau_1(G)$.*

Recall that any formula to which τ_2 is applied is a formula in a pre-clause form, and thus, has a structure $\mathbf{A}\Box(P \Rightarrow Q)$. Thus, we must ensure the transformation τ_2 has the following property:

Proposition 5. *For any ECTL formula Q if $\tau_2(\mathbf{A}\Box(P \Rightarrow Q))$ is satisfiable then so is $\mathbf{A}\Box(P \Rightarrow Q)$, where P is a literal or conjunction of literals.*

Proof. We prove this proposition by induction on the structure of Q.

For the base cases, (B1–B5 below) we have:

(B1) $Q = \mathbf{P}\Diamond l$ where $\mathbf{P}$ is either of path quantifiers and l is a literal. In this case

$$\tau_2\big(\mathbf{A}\Box(P \Rightarrow \mathbf{P}\Diamond l)\big) = \mathbf{A}\Box(P \Rightarrow \mathbf{P}\Diamond l)$$

Therefore, given

$$\langle \mathcal{M}, s_0\rangle \models \tau_2\big(\mathbf{A}\Box(P \Rightarrow \mathbf{P}\Diamond l)\big)$$

we immediately conclude that

$$\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{P}\Diamond l)$$

(B2) $Q = (l_1 \vee \cdots \vee l_n)$, where l_i $(1 \leqslant i \leqslant n)$ is a literal. In this case

$$\tau_2\big(\mathbf{A}\Box\big(P \Rightarrow (l_1 \vee \cdots \vee l_n)\big)\big) = \mathbf{A}\Box\big(\mathbf{start} \Rightarrow (\neg P \vee l_1 \vee \cdots \vee l_n)\big) \wedge \mathbf{A}\Box\big(\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee l_1 \vee \cdots \vee l_n)\big)$$

Thus, given

$$\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box\big(\mathbf{start} \Rightarrow (\neg P \vee l_1 \vee \cdots \vee l_n)\big) \wedge \mathbf{A}\Box\big(\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee l_1 \vee \cdots \vee l_n)\big)$$

we conclude that

$$\langle \mathcal{M}, s_0\rangle \models \big(\mathbf{start} \Rightarrow (\neg P \vee l_1 \vee \cdots \vee l_n)\big)$$
$$\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box\big(\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee l_1 \vee \cdots \vee l_n)\big)$$

(B3)–(B4) $Q = \mathbf{true}$ and $Q = \mathbf{false}$, respectively. Here the proof is immediate.

(B5) $Q = \mathbf{P}\bigcirc(l_1 \vee \cdots \vee l_n)$ where $\mathbf{P}$ is either of path quantifiers.
Here

$$\tau_2(\mathbf{A}\Box\big(P \Rightarrow \mathbf{P}\bigcirc(l_1 \vee \cdots \vee l_n)\big) = \mathbf{A}\Box\big(P \Rightarrow \mathbf{P}\bigcirc(l_1 \vee \cdots \vee l_n)\big)$$

Hence, given

$$\langle \mathcal{M}, s_0\rangle \models \tau_2\big(\mathbf{A}\Box\big(P \Rightarrow \mathbf{P}\bigcirc(l_1 \vee \cdots \vee l_n)\big)\big)$$

we immediately obtain

$$\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box\big(P \Rightarrow \mathbf{P}\bigcirc(l_1 \vee \cdots \vee l_n)\big)$$

Now, assuming as an induction hypothesis that Proposition 5 holds for any formula D_1, D_2, $\neg D_1$ and $\neg D_2$, we will show that it also holds for any of the following combinations: $D_1 \wedge D_2$, $\mathbf{P}\bigcirc(D_1 \wedge D_2)$, $\mathbf{PT}D_1$, $\mathbf{PT}^2 D_1$, $\mathbf{E}\Box\Diamond D_1$, $\mathbf{A}\Diamond\Box D_1$, etc.

Noting that proofs for $\mathbf{PT}D_1$, $\mathbf{P}D_1\mathbf{T}^2 D_2$ (where D_1 and D_2 are classical but not literals, **true** or **false**) and $\mathbf{P}\bigcirc(D_1 \wedge D_2)$ follow immediately from ECTL semantics, proof for $D_1 \wedge D_2$, which involves only classical reasoning, follows directly from $\mathrm{SNF}_{\mathrm{CTL}}$ semantics, and that proofs for the cases $\mathbf{E}\Box D_1$, $\mathbf{A}(D_1 \mathcal{U} D_2)$, $\mathbf{E}(D_1 \mathcal{U} D_2)$, $\mathbf{E}(D_1 \mathcal{W} D_2)$ and $\mathbf{A}(D_1 \mathcal{W} D_2)$ are similar to the case $(\mathbf{A}\Box D_1)$, we first consider the latter case and then two cases corresponding to the novel techniques introduced in this paper, namely, for $\mathbf{E}\Box\Diamond D_1$ and $\mathbf{A}\Box\Diamond D_1$.

Consider the case $Q = \mathbf{A}\Box D_1$. Here we are given

$$(\dagger) \quad \langle \mathcal{M}, s_0\rangle \models \tau_2\big(\mathbf{A}\Box(P \Rightarrow \mathbf{A}\Box D_1)\big)$$

from which we will show that

$$(\ddagger) \quad \langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{A}\Box D_1)$$

1. $\langle \mathcal{M}, s_0\rangle \models \tau_2(\mathbf{A}\Box(P \Rightarrow \mathbf{A}\Box y))$, from the condition (†);
2. $\langle \mathcal{M}, s_0\rangle \models \tau_2(\mathbf{A}\Box(y \Rightarrow D_1))$, from the condition (†).

Therefore, as τ_2 here is the '$\mathbf{A}\Box$' removal rule, from 1 we obtain 3 and 4:

3. $\langle \mathcal{M}, s_0\rangle \models \tau_2(\mathbf{A}\Box(P \Rightarrow (y \wedge z))$;
4. $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(z \Rightarrow \mathbf{A}\bigcirc y)$ and $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(z \Rightarrow \mathbf{A}\bigcirc z)$;

Now, following SNF_{CTL} semantics, we derive 5 and 6.

5. $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(\mathbf{start} \Rightarrow \neg P \vee y)$ and $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee y))$, from 3;
6. $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(\mathbf{start} \Rightarrow \neg P \vee z)$ and $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee z))$, from 3.
   By inductive hypothesis, from 2, we also have
7. $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(y \Rightarrow D_1)$.

Now we can see that $\mathcal{M}$ indeed satisfies $\mathbf{A}\Box(P \Rightarrow \mathbf{A}\Box D_1)$: from 5 we conclude that wherever P is satisfied in $\mathcal{M}$ so is y and hence, from 7, so is D_1. Also, 6 indicates that if P is satisfied at an arbitrary state s_i on an arbitrary fullpath χ then so is z. Hence, from 4, for any path π_{s_i}, a state s_{i+1}, the successor of s_i along π_{s_i}, satisfies both y and z. Therefore, from 7, s_{i+1} satisfies D_1. Again, as s_{i+1} satisfies z then any successor of s_{i+1} also satisfies y and z, hence, satisfies D_1, etc. Therefore, each path departing at s_i satisfies $\Box D_1$, i.e., we have shown that the conditions (‡) holds: $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{A}\Box D_1)$ as required.

Consider the case $Q = \mathbf{E}\Box\Diamond B$. Here we are given

$$(\dagger) \quad \langle \mathcal{M}, s_0\rangle \models \tau_2\big(\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Box\Diamond B_{\langle LC(\mathsf{ind})\rangle})\big)$$

from which we will show that

$$(\ddagger) \quad \langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{E}\Box\Diamond B_{\langle LC(\mathsf{ind})\rangle})$$

As here τ_2 is the application of the Renaming rule (the $\mathbf{E}\Box\Diamond$ case) we have

1. $\langle \mathcal{M}, s_0\rangle \models \tau_2(\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Box x_{\langle LC(\mathsf{ind})\rangle}))$, from the condition (†);
2. $\langle \mathcal{M}, s_0\rangle \models \tau_2(\mathbf{A}\Box(x \Rightarrow \mathbf{E}\Diamond B_{\langle LC(\mathsf{ind})\rangle}))$, from the condition (†);
   By inductive hypothesis we have
3. $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(x \Rightarrow \mathbf{E}\Diamond B_{\langle LC(\mathsf{ind})\rangle})$, from 2;

Now we can see that $\mathcal{M}$ indeed satisfies $\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Box\Diamond B_{\langle LC(\mathsf{ind})\rangle})$: from 1 we conclude that wherever P is satisfied in $\mathcal{M}$, from that point there exists a path associated with $\langle LC(\mathsf{ind})\rangle$ which satisfies $\Box x$. Also 3 indicates that if x is satisfied at an arbitrary state

s_i on an arbitrary fullpath χ then there must be path from s_i associated $\langle LC(\mathsf{ind})\rangle$ which satisfies $\Diamond B$. Therefore, we have shown that the conditions (‡) holds: $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{E}\Box\Diamond B_{\langle LC(\mathsf{ind})\rangle})$ as required.

Consider the case $Q = \mathbf{A}\Diamond\Box B$. Here we are given

$$(\dagger)\quad \begin{array}{l} \langle \mathcal{M}, s_0\rangle \models \tau_2\big[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_{1\,\langle LC(\mathsf{ind_1})\rangle})\big] \\ \langle \mathcal{M}, s_0\rangle \models \tau_2\big[\mathbf{A}\Box(x_1 \Rightarrow \mathbf{E}\Box B_{\langle LC(\mathsf{ind_1})\rangle})\big] \\ \ldots \\ \langle \mathcal{M}, s_0\rangle \models \tau_2\big[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_{n\,\langle LC(\mathsf{ind_n})\rangle})\big] \\ \langle \mathcal{M}, s_0\rangle \models \tau_2\big[\mathbf{A}\Box(x_n \Rightarrow \mathbf{E}\Box B_{\langle LC(\mathsf{ind_n})\rangle})\big] \end{array}$$

from which we will show that

$$(\ddagger)\quad \langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{A}\Diamond\Box B)$$

Recall that here τ_2 is the application of the Renaming rule (the $\mathbf{A}\Diamond\Box$ case). Let us abbreviate $\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_{1\,\langle LC(\mathsf{ind_1})\rangle})$, $\mathbf{A}\Box(x_1 \Rightarrow \mathbf{E}\Box B_{\langle LC(\mathsf{ind_1})\rangle}), \ldots, \mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_{n\,\langle LC(\mathsf{ind_n})\rangle})$, $\mathbf{A}\Box(x_n \Rightarrow \mathbf{E}\Box B_{\langle LC(\mathsf{ind_n})\rangle})$ as $a_1, \ldots, a_n$, respectively, and let $\mathcal{M}$ be a model which satisfies (†), where $\mathcal{M} = (S, R, L)$. Thus, we have $\langle \mathcal{M}, s_0\rangle \models a_1 \wedge \cdots \wedge a_n$. From Proposition 3, 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 it also satisfies ‡. The construction proceeds in the manner of [15]. We define our $(n+1)$-ary tree interpretation for formula † as $\langle \mathcal{M}', \lambda\rangle$, where $\mathcal{M}' = ([n+1]^\star, R, \pi)$ such that $\pi : [n+1]^\star \longrightarrow 2^{Prop}$ (see Definition 10), and inductively construct a mapping $\psi : [n+1]^* \longrightarrow S$, taking $\pi(s) = L(\psi(s))$.

We start by first selecting a linear interpretation $\langle \mathcal{M}_l, s_0\rangle$, where $\mathcal{M}_l = (S_l, R_l, L_l)$ from $\langle \mathcal{M}, s_0\rangle \models (a_1) \wedge \cdots \wedge (a_n)$ such that $\langle \mathcal{M}_l, s_0\rangle \models a_j$ $(1 \leqslant j \leqslant n)$, and define a mapping ψ_0 on the set $X_0 = 1^*$, as $\psi_0(1^k) = {R_l}^k(s_0)$ $(k \geqslant 0)$. This will be a basis path (labelled by φ) which is also referred to as the "leftmost" path of the canonical model in [15].

Now, given ψ_i defined on the set of nodes X_i, we define ψ_{i+1} and the set X_{i+1}. For each a_j from †, we choose a linear interpretation, $\langle \mathcal{M}_{l_j}, \psi(s)\rangle$ and define $\psi_{i+1}(s(j+1)) = R_{l_j}(\psi(s))$ and $\psi_{i+1}(s(j+1)1^k) = {R_{l_j}}^k(\psi(s))$ for $k \geqslant 0$. Finally, taking ψ to be ψ_ω completes the canonical model (see Fig. 5).

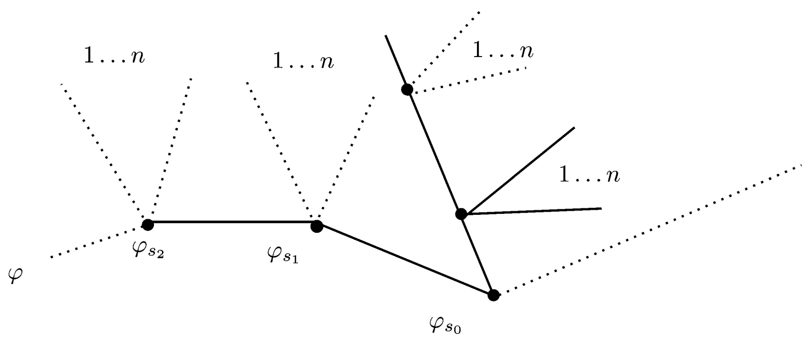

Fig. 5. $(n+1)$'ary canonical model for ‡.

The construction of $\mathcal{M}'$ ensures that $P \Rightarrow \mathbf{E}\Diamond\Box B$ is satisfied on every path from λ. Thus $\langle \mathcal{M}', \lambda\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{A}\Diamond\Box Q)$, from ECTL semantics. We also know that for any ECTL formula F, $\langle \mathcal{M}', \lambda\rangle \models F$ iff $\langle \mathcal{M}, s_0\rangle \models F$. Hence $\langle \mathcal{M}, s_0\rangle \models \mathbf{A}\Box(P \Rightarrow \mathbf{A}\Diamond\Box Q)$ (‡). □

Now from Lemmas 1, 2 and 3 we have the following theorem:

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

5. The temporal resolution method

Having provided the translation of ECTL formulae into $\mathrm{SNF}_{\mathrm{CTL}}$, 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

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

SRES 4

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

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 $\mathbf{start} \Rightarrow l$ and $\mathbf{start} \Rightarrow \neg l$, will be $\mathbf{start} \Rightarrow \mathbf{false}$.

Temporal resolution rules. The basic idea of invoking temporal resolution is to resolve a clause containing $\Diamond\neg l$ together with a set of formulae characterizing a *loop* in l, a set of $\mathrm{SNF}_{\mathrm{CTL}}$ 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 [5].

TRES 1

$$\frac{P \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\Box l \qquad Q \Rightarrow \mathbf{A}\Diamond\neg l}{Q \Rightarrow \mathbf{A}(\neg P \mathcal{W} \neg l)}$$

TRES 2

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

TRES 3

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

TRES 4

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

where the first premise is the abbreviation for the **A** loop or **E** loop in l given that P is satisfied.

The aim of applying the temporal resolution method to a set of SNF_{CTL} clauses, R, is to derive an empty clause **start** $\Rightarrow$ **false**. If we have derived an empty clause then the procedure terminates and the set R is unsatisfiable. If we have not achieved this applying SRES rules and there is no eventuality clause in the set of clauses then the procedure terminates indicating that R is satisfiable. Alternatively, we create a list of eventualities, say $l_1, \ldots, l_n$ and start looking for loops in $\neg l_1, \neg l_2, \ldots$ which will lead us to the application of the corresponding TRES rule and subsequent chain of transformations and further applications of SRES. This chain repeats until we either derive an empty clause or no more rules have become applicable. (For full details of the method see [5,6].)

Correctness of the transformation of ECTL formulae into SNF_{CTL} (Section 4.4) together with the termination and correctness of the resolution method defined over SNF_{CTL} (shown in [2,6]) enables us to apply the latter as the refutation method for ECTL. Namely, given an ECTL formula G, translate $\neg G$ into $\text{SNF}_{\text{CTL}}(G)$ and apply the temporal resolution method to the latter. If an empty clause is derived then $\neg G$ is unsatisfiable, hence G is valid, otherwise $\neg G$ is satisfiable, hence G is not valid.

Example refutation. We apply the resolution method to the set of SNF_{CTL} clauses obtained for ECTL formula $\mathbf{A}\Diamond\Box p \Rightarrow \mathbf{A}\Diamond p$ (formula (6) in section Section 4.3). We commence the proof presenting at steps 1–8 only those clauses that are involved into the resolution refutation in the following order: initial clauses, step clauses and, finally, any sometime clauses.

1. **start** $\Rightarrow x$
2. **start** $\Rightarrow \neg x \vee y$
3. **start** $\Rightarrow \neg x \vee \neg p$
4. **start** $\Rightarrow \neg l \vee p$
5. **true** $\Rightarrow \mathbf{A}\bigcirc(\neg l \vee p)$
6. $y \Rightarrow \mathbf{E}\bigcirc\neg p_{\langle \mathsf{f}\rangle}$
7. $y \Rightarrow \mathbf{E}\bigcirc y_{\langle \mathsf{f}\rangle}$
8. $x \Rightarrow \mathbf{E}\Diamond l_{\langle LC(\mathsf{f})\rangle}$

As we mentioned, the first stage of applying the temporal resolution method is the application of SRES rules in order to either derive an empty clause or to specify a set of step clauses for a loop searching technique. Thus, applying step resolution rules we obtain steps 9–12.

9.	$\mathbf{start} \Rightarrow \neg p$	$1, 3$	*SRES 1*
10.	$\mathbf{start} \Rightarrow y$	$1, 2$	*SRES 1*
11.	$\mathbf{start} \Rightarrow \neg l$	$4, 9$	*SRES 1*
12.	$y \Rightarrow \mathbf{E}\bigcirc\neg l_{\langle \mathsf{f} \rangle}$	$5, 6$	*SRES 2*

As we have not derived an empty clause and there is an eventuality clause 8, we are looking for a loop in $\neg l$. The desired loop is given by clauses 7 and 12, namely these formulae together represent a $\mathbf{E}\Box$ loop in $\neg l$: $y \Rightarrow \mathbf{E}\Box\mathbf{E}\bigcirc\neg l_{\langle LC(\mathsf{f})\rangle}$. Thus, we apply the TRES 4 rule to resolve this loop and clause 8, obtaining 13.

13. $x \Rightarrow \mathbf{E}(\neg y \mathcal{W} l)_{\langle LC(\mathsf{f})\rangle}$ 7, 12, 8 *TRES 4*

At this stage we remove $\mathbf{E}\mathcal{W}$, and use only one of the conclusions of this rule. This gives us a purely classical formula on step 14 below, where z is a new variable.

14. $x \Rightarrow l \vee \neg y \wedge z$ 13, Removal of $\mathbf{E}\mathcal{W}$

Now, applying some classical transformations together with the temporising rule, we derive 15, and finally, a chain of applications of the SRES 1 gives us the terminating clause $\mathbf{start} \Rightarrow \mathbf{false}$.

15.	$\mathbf{start} \Rightarrow \neg x \vee l \vee \neg y$	14, *classical*, *Temporizing*
16.	$\mathbf{start} \Rightarrow \mathbf{false}$	1, 10, 11, 15 *SRES 1*

6. Conclusions and future work

We have described the extension of the clausal resolution method to the useful branching-time logic ECTL. To the best of our knowledge there are no analogous clausal resolution methods developed for branching-time logics. One of the obvious benefits of using the clausal resolution technique is the possibility of invoking a variety of well-developed methods and refinements used in the framework of classical logic. Another obvious advantage of using this technique is its capacity to handle arbitrary systems, while other methods, such as for example, the model checking technique [1], are restricted to the analysis of finite state systems.

Note also, that in [7], it was shown that the normal form developed for linear-time temporal logic is as expressive as Büchi word automata, and, therefore, as propositional linear-time μ-calculus [8]. Thus, in the linear-time case we are able to represent a problem specification directly as a set of formulae in the normal form and apply a resolution based verification technique to the latter. This opens another direction of our future work—analysis of the correspondence of $\mathrm{SNF}_{\mathrm{CTL}}$, the normal form for several branching-time logics, CTL, ECTL, and ECTL^{+}, and automata for branching-time logics.

The algorithm to search for loops needed for temporal resolution was 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. Taking into account these observations, we define a future task to *refine* this algorithm, to analyse the *complexity* of the clausal resolution method for logics CTL, ECTL and ECTL^{+}, and to develop corresponding prototype systems.

We believe that a number of techniques explored in this paper will be useful in developing the resolution method for other extensions of CTL culminating in CTL^{*}:

(1) The method of identifying different types of nesting of temporal operators understood as minimal or maximal fixpoints. We have shown that in the 'bad' nesting, a temporal operator defined as a maximal fixpoint is prefixed by an '**E**' quantifier or a temporal operator defined as a minimal fixpoint is prefixed by an '**A**' quantifier.

(2) Our novel technique to transform ECTL formulae which contain fairness constraints representing these 'bad' cases of nesting of temporal operators is based upon the indexing and the existence of the canonical model. Note that the canonical model construction was crucial in our correctness argument.

(3) The technique of analysing formulae which have some structural similarity but have different satisfiability characteristics. For example, a 'tiny' change of the satisfiable $\mathrm{CTL}^{\star}$ formula $\mathbf{A}\Diamond(\bigcirc p \wedge \mathbf{E}\bigcirc\neg p)$ to $\mathbf{A}\Diamond(\mathbf{E}\bigcirc p \wedge \mathbf{E}\bigcirc\neg p)$ makes the latter unsatisfiable. Thus, in developing the required transformation rules it will be useful to have a test-bench of such $\mathrm{CTL}^{\star}$ formulae which will also be an effective method of testing the correlation of the transformation rules under development and the desired resolution procedure.

Acknowledgements

We would like to thank Michael Fisher, Clare Dixon and Mark Reynolds for useful discussions during the preliminary work on the paper, and Renate Schmidt for valuable comments on the relevant material in the PhD thesis [2]. We are also grateful to anonymous referees for their suggestions on improving this article.

References

[1] O. Bernholtz, M.Y. Vardi, P. Wolper, An automata-theoretic approach to branching-time model checking, in: Computer Aided Verification, Proc. 6th Int. Workshop, Stanford, CA, in: Lecture Notes in Computer Science, vol. 818, Springer, Berlin, 1994, pp. 142–155.

[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, 2003.

[4] A. Bolotov, A. Basukoski, A clausal resolution for branching-time logic ECTL^{+}, in: Proceedings of the Time-2004, IEEE, 2004, pp. 140–147.

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

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

[7] A. Bolotov, M. Fisher, C. Dixon, On the relationship between 'w'-automata and temporal logic normal form, J. Logic Comput. 12 (2002) 561–581.

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

[9] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronisation skeletons using branching time temporal logic, in: Logic of Programs, Proceedings of Workshop, in: Lecture Notes in Computer Science, vol. 131, Springer, Berlin, 1981, pp. 52–71.

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

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

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

[13] E.A. Emerson, A.P. Sistla, Deciding full branching time logic, in: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC 1984), 1984, pp. 14–24.

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

[15] P. Wolper, On the relation of programs and computations to models of temporal logic, in: L. Bolc, A. Szałas (Eds.), Time and Logic, a Computational Approach, UCL Press Limited, 1995, pp. 131–178, Chapter 3.