A clausal resolution for extended computation tree logic ECTL.

Alexander Bolotov
Harrow School of Computer Science

A Clausal Resolution Method for Extended Computation Tree Logic ECTL

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

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.*

1 Introduction

A Computation Tree Logic (CTL), first proposed in [6], and its extensions have shown to play a significant role in potential applications [8]. CTL does not permit boolean combinations of formulae with temporal operators or their nesting. Two combinations of future time temporal operators $\diamondsuit$ ('sometime') and $\Box$ ('always'), are useful in expressing *fairness* [7]: $\diamondsuit\Box p$ (p is true along the path of the computation except possible some finite initial interval of it) and $\Box\diamondsuit p$ (p is true along the computation path at infinitely many moments of time).

The logic ECTL (Extended CTL [9]) 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 [2, 3] a clausal resolution approach to CTL has been developed, extending the original definition of the method for the linear-time case [11]. 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. As a normal form for ECTL we utilise the Separated Normal Form developed for CTL formulae, called $\mathrm{SNF}_{\mathrm{CTL}}$. This enables us to apply the resolution technique defined over $\mathrm{SNF}_{\mathrm{CTL}}$ 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 $\mathrm{SNF}_{\mathrm{CTL}}$ by *a novel* transformation technique to cope with ECTL fairness. $\mathrm{SNF}_{\mathrm{CTL}}$ 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 [4], 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 §2 we outline the syntax and semantics of ECTL and those properties of ECTL syntax and semantics that are important for our analysis. In §3 we review $\mathrm{SNF}_{\mathrm{CTL}}$. The translation algorithm, novel transformation technique to cope with fairness as well and main rules, which are used in the example transformation, are given in §4. We conclude this section providing an example and the correctness argument. In §5 we outline the temporal resolution method defined over $\mathrm{SNF}_{\mathrm{CTL}}$ and apply it to a set of $\mathrm{SNF}_{\mathrm{CTL}}$ clauses (previously obtained in §4.2). Finally, in §6, we draw conclusions and discuss future work.

Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and
Fourth International Conference on Temporal Logic (TIME-ICTL'03) 1530-1311/03 $17.00 © 2003 IEEE

2 Syntax and Semantics of ECTL

In the language of ECTL we extend the language of linear-time temporal logic, which uses future time $\Box$ (always), $\Diamond$ (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 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)

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

Examples of ECTL formulae are $\mathbf{A}\Diamond\Box A$, $\mathbf{A}\Box\Diamond A$, $\mathbf{E}\Diamond\Box A$ and $\mathbf{E}\Box\Diamond A$ (where A is any ECTL formula), which express the fairness properties.

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.

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 $\chi_{s_i}, \langle \mathcal{M}, \chi_{s_i}\rangle \models B$.
$\langle \mathcal{M}, s_i\rangle \models \mathbf{E}B$ 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 $i \leq j$ then $\langle \mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$.
$\langle \mathcal{M}, \chi_{s_i}\rangle \models \Diamond B$ iff there exists $s_j \in \chi_{s_i}$ such that $i \leq j$ and $\langle \mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$.
$\langle \mathcal{M}, \chi_{s_i}\rangle \models \bigcirc B$ 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 there exists $s_j \in \chi_{s_i}$ such that $i \leq j$ and $\langle \mathcal{M}, Suf(\chi_{s_i}, s_j)\rangle \models B$ and for each $s_k \in \chi_{s_i}$, if $i \leq k < j$ then $\langle \mathcal{M}, Suf(\chi_{s_i}, s_k)\rangle \models A$.
$\langle \mathcal{M}, \chi_{s_i}\rangle \models A\,\mathcal{W}\,B$ 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 *[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 2 *[Validity] A well-formed ECTL formula, B, is valid if, and only if, it is satisfied in every possible model.*

2.1 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. 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 follows a computation for an arbitrarily long time, then it can follow a computation forever. This corresponds to limit closure property, meaning that for any fullpath γ_{s_0} and any paths $\pi_{s_j}, \varphi_{s_k}, \ldots$ such that γ_{s_0} has the prefix $[s_0, s_j]$, π_{s_j} has the prefix $[s_j, s_k]$, φ_{s_k} has the prefix $[s_k, s_l]$, etc, and $0 < j < k < l$, the following holds (see Figure 1): there exists an infinite path α_{s_0} that is a limit of the prefixes $[s_0, s_j], [s_j, s_k], [s_k, s_l], \ldots$.

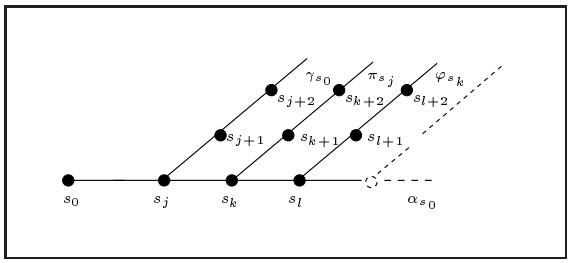

Figure 1. Limit closure

We assume that tree-like models of ECTL are suffix, fusion and limit closed.

2.2 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.

Fairness Constraints. Validity of the following equivalences can be easily shown:

$$\mathbf{A}\Box\Diamond B \equiv \mathbf{A}\Box\mathbf{A}\Diamond B,\ \ \mathbf{E}\Diamond\Box B \equiv \mathbf{E}\Diamond\mathbf{E}\Box B \qquad (1)$$

Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and Fourth International Conference on Temporal Logic (TIME-ICTL'03) 1530-1311/03 $17.00 © 2003 IEEE

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 [7]. 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 would significantly affect the renaming of the embedded paths subformulae in the corresponding ECTL fairness constraints.

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

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

A model, $\mathcal{M}$, for this formula (see Figure 1) 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 $Suf(\alpha_{s_0}, s_j)$ and also at $Suf(\gamma_{s_0}, s_{j+2})$, $Suf(\pi_{s_j}, s_{k+2})$, $Suf(\varphi_{s_k}, s_{l+2}), \ldots$. Finally, let s_{j+1}, s_{k+1}, $s_{l+1}, \ldots$ along paths γ, π, $\varphi, \ldots$, respectively, satisfy $\neg p$.
Note that if we change the first conjunct of formula (2) to $\mathbf{A}\Diamond\mathbf{A}\Box p$ then the whole formula becomes unsatisfiable.

Notation.

- In the rest of the paper, let **T** abbreviate any unary and $\mathbf{T}^2$ any binary temporal operator and **P** either of path quantifiers.
- Any formula of the type **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 constrains, $\mathbf{P}\Box\Diamond$ and $\mathbf{P}\Diamond\Box$.
- 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.
- A *literal* is a proposition or its negations.

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 3 (Degree of path quantifier nesting)

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

Emerson and Sistla [10] showed that any CTL* formula F can be transformed into F' such that $N(F') \leq 2$. 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 4 (Reduction of the path quantifier nesting) *Given an ECTL formula G such that $N(G) \geq 2$, the following procedure reduces the nesting of path quantifiers in G to the degree 2: $Red[G] = \mathbf{A}\Box(x_1 \equiv S_1) \wedge Red[G(S_1/x_1)]$, where S_1 is the designated* **P**-*embedded state subformula of G, x_1 is a new proposition and $G(S_1/x_1)$ is a result of the replacement of S_1 in G by x_1. If $N(G) < 2$ then the procedure terminates.*

For example, given $G = \mathbf{A}\Diamond(\mathbf{E}\Box\Diamond\neg p \wedge \mathbf{A}\Diamond\mathbf{A}\Box p)$ we can obtain $Red[G] = \mathbf{A}\Box(x_1 \equiv \mathbf{A}\Box p) \wedge \mathbf{A}\Box(x_2 \equiv \mathbf{A}\Diamond x_1) \wedge \mathbf{A}\Box(x_3 \equiv \mathbf{E}\Box\Diamond\neg p) \wedge \mathbf{A}\Diamond(x_3 \wedge x_2)$.

Proposition 1 [Correctness of the Reduction procedure] *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 4* [10].

Negation Normal Form for ECTL. Using the standard technique we can translate an ECTL formula G into its negation normal form, $\mathrm{NNF}_{\mathrm{ECTL}}(G)$ [7].

Proposition 2 [Correctness of $\mathbf{NNF_{ECTL}}$] *For any* ECTL *formula, G, the following holds:* $\langle \mathcal{M}, s_0\rangle \models G$ iff $\langle \mathcal{M}, s_0\rangle \models \mathrm{NNF}_{\mathrm{ECTL}}(G)$.

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

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

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

Branching Factor. Below 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 5 (Branching degree of a state) *The number of immediate successors of a state s in a tree structure is called a* branching degree *of s.*

Definition 6 (Branching factor of a tree structure) *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 tree models are of at most countable branching. However, following ([7], page 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".

Now, following [12], given that an ECTL model structure $\mathcal{M}$ has its branching factor at most k, there exists a k-ary tree canonical model $\mathcal{M}'$ such that for any formula F, $\mathcal{M}$ satisfies F if, and only if, $\mathcal{M}'$ satisfies F. Informally, a canonical model is an unwinding of an arbitrary model $\mathcal{M}$ into an infinite tree $\mathcal{T}$ [12].

Definition 7 (Tree canonical model) *Let $\mathcal{T} = (N, E)$ be a k-ary infinite tree such that $[k]$ denotes the set $\{1, \ldots k\}$, and*

- *$N = [k]^\star$ is a set of states, with the root being an empty string λ*
- *$E = \{(s, s_i) | s \in [k]^\star,\ i \in [k]\}$, where s_i $(i \in [k])$ is a set of successors of a state s.*

Now, given an alphabet $\Sigma = 2^{Prop}$, a canonical tree model *for ECTL is of the form $\langle \mathcal{M}, \lambda \rangle$, where $\mathcal{M} = ([k]^\star, E, L)$ such that $L : [k]^\star \longrightarrow 2^{Prop}$ is a function which assigns truth values to the atomic propositions in each state.*

Proposition 3 given below collects the results given in [12] (Lemma 3.4 and Lemma 3.5, pages 144-145).

Proposition 3 (Existence of a canonical model for ECTL)

- *If an ECTL formula F has a model M whose branching factor is $\leq k$ then F has a tree canonical model $\langle \mathcal{M}, \lambda \rangle$, where $\mathcal{M} = ([k]^\star, \{(s, s_i) | s \in [k]^\star,\ i \in [k]\}, L)$.*
- *If an ECTL formula F containing n (existential) path quantifiers has a model, then it has an $(n+1)$-ary canonical model.*

We will essentially use these results for the formulation of the transformation rule managing ECTL fairness constraints, namely, formulae that contain $\mathbf{A}\Diamond\Box$.

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 [1, 3]. Identifying the core operators, $\mathbf{P}\bigcirc$ and $\mathbf{P}\Diamond$, we are able to generate formulae relevant to either the first state in a model, or to all subsequent states in a model. Transforming ECTL formulae into $\mathrm{SNF}_{\mathrm{CTL}}$ we aim to remove all other, unwanted, modalities $\mathbf{A}\Box, \mathbf{A}\,\mathcal{U}, \ldots$. 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 $\langle \mathsf{LC(ind)} \rangle$ which represents 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.

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 $\mathbf{true}$ and $\mathbf{false}$, and a new operator, $\mathbf{start}$ ('at the initial moment of time') defined as $\langle \mathcal{M}, s_i \rangle \models \mathbf{start} \quad \text{iff} \quad i = 0$.

Definition 8 (Separated Normal Form $\mathrm{SNF}_{\mathrm{CTL}}$) $\mathrm{SNF}_{\mathrm{CTL}}$ *is a set of formulae*

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

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

$$\begin{array}{lcll}
\mathbf{start} & \Rightarrow & \bigvee_{j=1}^{k} \beta_j & \text{an Initial Clause} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{A}\bigcirc\left[\bigvee_{j=1}^{k} \beta_j\right] & \text{an } \mathbf{A} \text{ step clause} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{E}\bigcirc\left[\bigvee_{j=1}^{k} \beta_j\right]_{\langle \mathsf{ind} \rangle} & \text{a } \mathbf{E} \text{ step clause} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{A}\Diamond\gamma & \text{an } \mathbf{A} \text{ sometime clause} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{E}\Diamond\gamma_{\langle \mathsf{LC(ind)} \rangle} & \text{a } \mathbf{E} \text{ sometime clause}
\end{array}$$

Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and
Fourth International Conference on Temporal Logic (TIME-ICTL'03) 1530-1311/03 $17.00 © 2003 IEEE

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

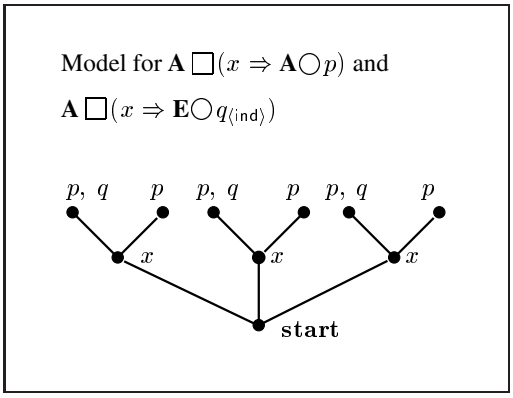

Figure 2. Interpretation of step clauses.

Thus, a clause $\mathbf{A}\Box(x \Rightarrow \mathbf{A}\bigcirc p)$ (a model for which is given in Figure 2) is interpreted as "*for any fullpath χ and any state $s_i \in \chi$ $(0 \leq i)$, 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*".

A clause $\mathbf{A}\Box(x \Rightarrow \mathbf{E}\bigcirc q_{\langle \mathsf{ind} \rangle})$ (see Figure 2) is understood as "*for any fullpath χ and any state $s_i \in \chi$ $(0 \leq i)$, if x is satisfied at a state s_i then q must be satisfied at the moment, next to s_i, along some path associated with $\langle \mathsf{ind} \rangle$ which departs from s_i*".

Finally, $\mathbf{A}\Box(x \Rightarrow \mathbf{E}\Diamond p_{\langle \mathsf{LC}(\mathsf{ind}) \rangle})$ (see Figure 3) has the following meaning "*for any fullpath χ and any state $s_i \in \chi$ $(0 \leq i)$, if x is satisfied at a state s_i then p must be satisfied at some state, say s_j $(i \leq j)$, along some path α_{s_i} associated with the limit closure of $\langle \mathsf{ind} \rangle$ which departs from s_i*".

4 Transformation of ECTL formulae into $\mathbf{SNF_{CTL}}$

As $\mathrm{SNF_{CTL}}$ 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 $C = \mathrm{NNF_{ECTL}}(\neg G)$. 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 $\mathbf{start}$ and apply the initial renaming rule obtaining $\mathbf{A}\Box(\mathbf{start} \Rightarrow x_0) \wedge \mathbf{A}\Box(x_0 \Rightarrow C)$, where x_0 is a new proposition.

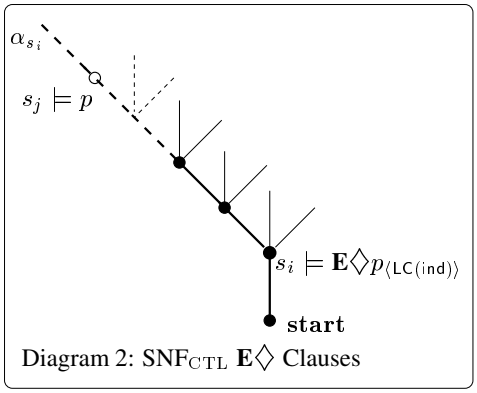

Diagram 2: $\mathrm{SNF_{CTL}}$ $\mathbf{E}\Diamond$ Clauses

Figure 3. Interpretation of sometime clauses.

2. Apply equations (1) and then procedure Red (see Definition 4) 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 $\mathbf{start}$, 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 an unique index $\langle \mathsf{ind}_\mathsf{i} \rangle \in \mathsf{IND}$ and any other pre-clause containing the $\mathbf{E}$ quantifier by an unique index $\langle \mathsf{LC}(\mathsf{ind}_\mathsf{j}) \rangle \in \mathsf{IND}$. 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.
7. Derive the desired form of $\mathrm{SNF}_{\mathrm{CTL}}$ 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 $\mathbf{P}\bigcirc$ 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 (see below).

4.1 Transformation rules towards $\mathrm{SNF}_{\mathrm{CTL}}$

In the transformation procedure τ outlined above, the first stage, the procedure τ_1, except for the application of equations (1) at step 2, is taken from the translation of CTL formulae to $\mathrm{SNF}_{\mathrm{CTL}}$ [1]. 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 §4.2. For the full set of rules preserved from the CTL the reader is referred to [1, 3].
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 $\mathrm{SNF}_{\mathrm{CTL}}$ pre-clauses containing the **E** quantifier: here we first label every pre-clause $P \Rightarrow \mathbf{E}\bigcirc Q$ by an 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 **E** quantifier we associate an 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 equations (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:

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

where we introduce a new 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 satisifed 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 $s_k \in \chi\,(0 \leq k)$,
> 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 Figure 4).

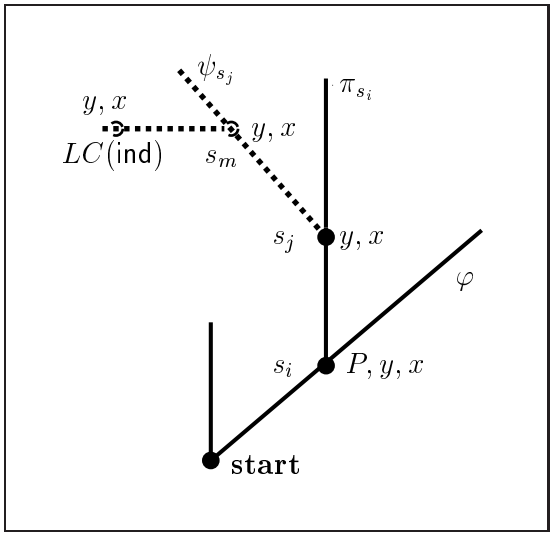

Figure 4. Labelling ECTL formulae: the LC index

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

$$\langle \mathcal{M}', \chi \rangle \models \Box(P \Rightarrow (y \wedge x)),$$

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

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, we define a

model $\mathcal{M}'$ to be the same as $\mathcal{M}$ except for a new proposition x such that for a path π_{s_i} (associated with $\langle \mathsf{ind} \rangle$), for any state $s_n \in \pi_{s_i}$, if $n \leq i$ 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 $y \wedge x$. 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}. 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 §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 equation (5) to remove the $\mathbf{E}\Box$ modality as follows (in formulation of the rules below x is a new proposition):

Removal of $\mathbf{E}\Box$

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

Other removal rules for basic CTL modalities are:

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

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

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

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

Managing embedded path subformulae in ECTL. 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 §2.2). 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.

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

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 $\mathbf{E}$ pre-clauses. Now we use this information about the number of existential path quantifiers based upon proof of Proposition 3 [12], 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."

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 $\langle LC(\mathsf{ind}) \rangle$ (which can be easily justified).

Now, based on Proposition 3, we rename the $\Box Q$ subformula of $\mathbf{A}\Diamond\Box Q$ as follows.

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

if $n = 0$

$$\begin{array}{lcl} P & \Rightarrow & \mathbf{A}\Diamond\Box Q \\ \hline P & \Rightarrow & \mathbf{E}\Diamond x_{\langle LC(\mathsf{ind}) \rangle} \\ 1 & \Rightarrow & \mathbf{E}\Box Q_{\langle LC(\mathsf{ind}) \rangle} \end{array}$$

if $n > 0$

$$\begin{array}{lcl} P & \Rightarrow & \mathbf{A}\Diamond\Box Q \\ \hline P & \Rightarrow & \mathbf{E}\Diamond x_{1\langle LC(\mathsf{ind}_1) \rangle} \\ x_1 & \Rightarrow & \mathbf{E}\Box Q_{\langle LC(\mathsf{ind}_1) \rangle} \\ & \cdots & \\ P & \Rightarrow & \mathbf{E}\Diamond x_{n\langle LC(\mathsf{ind}_n) \rangle} \\ x_n & \Rightarrow & \mathbf{E}\Box Q_{\langle LC(\mathsf{ind}_n) \rangle} \end{array}$$

where n is the number of indices in *LIST_IND* and $x, x_1, \ldots, x_n$ are new propositions.

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

Temporising

$$\frac{Q \Rightarrow P}{\mathbf{start} \Rightarrow \neg P \vee Q \quad \mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg P \vee \neg Q)}$$

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.

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

$$\frac{P \Rightarrow \mathbf{A}\bigcirc(P \wedge Q)}{P \Rightarrow \mathbf{A}\bigcirc P \quad P \Rightarrow \mathbf{A}\bigcirc Q}$$

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

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.

4.2 Example Transformation

As an example we translate into $\mathrm{SNF_{CTL}}$ the following ECTL validity:

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

To check that (6) is valid we negate it, obtaining $\neg(\mathbf{A}\diamondsuit\Box p \Rightarrow \mathbf{A}\diamondsuit p)$ and derive the *Negation Normal Form* of (6), $\mathbf{A}\diamondsuit\Box p \wedge \mathbf{E}\Box\neg p$. Following the translation algorithm, we derive steps 0–2, where x is a new proposition, and split conjunction on the right hand side of the formula at step 2, obtaining steps 3–4.

0.	$\mathbf{start}$	$\Rightarrow$	$\mathbf{A}\diamondsuit\Box p \wedge \mathbf{E}\Box\neg p$	anchoring to $\mathbf{start}$
1.	$\mathbf{start}$	$\Rightarrow$	x	0, Initial Renaming
2.	x	$\Rightarrow$	$\mathbf{A}\diamondsuit\Box p \wedge \mathbf{E}\Box\neg p$	0, Initial Renaming
3.	x	$\Rightarrow$	$\mathbf{A}\diamondsuit\Box p$	from 2, classical
4.	x	$\Rightarrow$	$\mathbf{E}\Box\neg p$	from 2, classical

At this stage we first label pre-clause 4 by a new label, $\langle \mathsf{f}\rangle$ and then rename $\Box p$ in 3, introducing a new variable, l.

5.	x	$\Rightarrow$	$\mathbf{E}\diamondsuit l_{\langle LC(\mathsf{f})\rangle}$	from 3
6.	l	$\Rightarrow$	$\mathbf{E}\Box p_{\langle LC(\mathsf{f})\rangle}$	from 3

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

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

Now note that steps 7 and 9 are purely classical expressions. Here, first splitting conjunctions on the right hand side of these formulae, and then introducing a temporal context incorporating the rule *Temporising*, we derive the steps below:

11.	$\mathbf{start}$	$\Rightarrow$	$\neg x \vee \neg p$	from 7
12.	$\mathbf{true}$	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg x \vee \neg p)$	from 7
13.	$\mathbf{start}$	$\Rightarrow$	$\neg x \vee y$	from 7
14.	$\mathbf{true}$	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg x \vee y)$	from 7
15.	$\mathbf{start}$	$\Rightarrow$	$\neg l \vee p$	from 9
16.	$\mathbf{true}$	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg l \vee p)$	from 9
17.	$\mathbf{start}$	$\Rightarrow$	$\neg l \vee r$	from 9
18.	$\mathbf{true}$	$\Rightarrow$	$\mathbf{A}\bigcirc(\neg l \vee r)$	from 9

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

19.	y	$\Rightarrow$	$\mathbf{E}\bigcirc\neg p_{\langle \mathsf{f}\rangle}$	from 8
20.	y	$\Rightarrow$	$\mathbf{E}\bigcirc y_{\langle \mathsf{f}\rangle}$	from 8
21.	r	$\Rightarrow$	$\mathbf{E}\bigcirc p_{\langle \mathsf{f}\rangle}$	from 10
22.	r	$\Rightarrow$	$\mathbf{E}\bigcirc r_{\langle \mathsf{f}\rangle}$	from 10

The normal form of the given ECTL formula is represented by clauses 1, 5, 11–22.

4.3 Correctness of the Transformation of ECTL formulae into $\mathbf{SNF_{CTL}}$

We first show that an ECTL formula G is satisfiable, if and only if, $\tau_1(F)$ is satisfiable (Lemma 1). Next, we will establish that the transformation procedure τ_2 preserves satisfiability (Lemma 2).

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

PROOF: Since procedure τ_1 is taken from the translation of CTL formulae to $\mathrm{SNF_{CTL}}$, proof of Lemma 1 simply repeats stages of the corresponding proof for CTL [1], taking into account Proposition 1, Proposition 2, and equivalences (1). (END)

Lemma 2 *Given a $\mathrm{SNF_{CTL}}$ 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 $\mathbf{start}$, Q_j is either a purely classical formula or $Q_j = \mathbf{PT}C_j$ or $Q_j = \mathbf{P}\Box\diamondsuit C_j$ or $Q_j = \mathbf{P}\diamondsuit\Box C_j$ 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, then we must consider the cases, corresponding to possible applications of $\tau_2[\mathbf{A}\Box S]$. These cases correspond to the stages 3–7 of the transformation algorithm described in §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 again repeats stages of the corresponding proof for CTL [1].

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

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

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

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

In the corresponding proof we obtain a model $\mathcal{M}'$ from $\mathcal{M}$ by letting a new 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.

Case 2. Here we apply τ_2 in the following way

$$\frac{\tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{A}\Diamond\Box Q)]}{\begin{array}{l}\tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_1)_{\langle LC(\mathsf{ind}_1)\rangle}] \\ \tau_2[\mathbf{A}\Box(x_1 \Rightarrow \mathbf{E}\Box Q)_{\langle LC(\mathsf{ind}_1)\rangle}] \\ \ldots \\ \tau_2[\mathbf{A}\Box(P \Rightarrow \mathbf{E}\Diamond x_n)_{\langle LC(\mathsf{ind}_n)\rangle}] \\ \tau_2[\mathbf{A}\Box(x_n \Rightarrow \mathbf{E}\Box Q)_{\langle LC(\mathsf{ind}_n)\rangle}]\end{array}}$$

Again, the corresponding proof shows that given a model $\mathcal{M}$ which satisfies the condition of Proposition 4 in this case, there exists a model $\mathcal{M}'$ which satisfies its conclusions. Here, we derive $\mathcal{M}'$ from $\mathcal{M}$ based upon Proposition 3, first labelling paths of $\mathcal{M}'$ by the indices $\langle LC(\mathsf{ind}_1)\rangle, \ldots, \langle LC(\mathsf{ind}_n)\rangle$ and then by relevant labelling of the states of $\mathcal{M}$ by new propositions $x_1, x_2, \ldots, x_n$. The result follows taking into account the interpretation of the **E** clauses labelled with the 'LC' type indices. 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. (END)

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 [1, 3]: *step* resolution (SRES) and *temporal* resolution (TRES). Here we give only those step and temporal resolution rules which are used in the example refutation. For a detailed description of the resolution technique defined over $\mathrm{SNF}_{\mathrm{CTL}}$ see [1, 3].

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.

SRES 1

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

SRES 2

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

Temporal Resolution Rules. The basic idea of invoking temporal resolution is to resolve 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, together with the clause containing $\Diamond\neg l$ [2]. Below we formulate the TRES 4 rule.

TRES 4

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

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

Correctness of the transformation of ECTL formulae into $\mathrm{SNF}_{\mathrm{CTL}}$ (§4.3) together with the termination and correctness of the resolution method defined over $\mathrm{SNF}_{\mathrm{CTL}}$ (shown in [1, 3]) enables us to apply the latter as the refutation method for ECTL.

Example Refutation. We apply the resolution method to the set of $\mathrm{SNF}_{\mathrm{CTL}}$ clauses obtained for ECTL formula $\mathbf{A}\Diamond\Box p \Rightarrow \mathbf{A}\Diamond p$ (formula (6) in section §4.2). 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. $\mathbf{start} \Rightarrow x$
2. $\mathbf{start} \Rightarrow \neg x \vee y$
3. $\mathbf{start} \Rightarrow \neg x \vee \neg p$
4. $\mathbf{start} \Rightarrow \neg l \vee p$
5. $\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg l \vee p)$
6. $y \Rightarrow \mathbf{E}\bigcirc\neg p_{\langle \mathsf{f}\rangle}$
7. $y \Rightarrow \mathbf{E}\bigcirc y_{\langle \mathsf{f}\rangle}$
8. $x \Rightarrow \mathbf{E}\Diamond l_{\langle LC(\mathsf{f})\rangle}$

Now, 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 clauses 7 and 12 represent a $\mathbf{E}\Box$ loop in $\neg l$: $y \Rightarrow \mathbf{E}\Box\mathbf{E}\bigcirc\neg l_{\langle LC(\mathsf{f})\rangle}$, 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, Temp.*
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. 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. The algorithm to search for loops needed for temporal resolution has been introduced in [2]. 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, and having analysed the *complexity* of the clausal resolution method for both logics, CTL and ECTL, 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 the extensions of ECTL to ECTL^{+} and 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 a '**E**' quantifier or a temporal operator defined as a minimal fixpoint is prefixed by a '**A**' quantifier.
(2) The technique of analysing formulae which have some structural similarity but have different satisfiability characteristics. For example, a 'tiny' change of the $\mathrm{CTL}^{\star}$ 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 ECTL^{+} and $\mathrm{CTL}^{\star}$ 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, Renate Schmidt for valuable comments on the relevant material in the author's PhD thesis, and anonymous referees for corrections and suggestions on improving the quality of this paper.

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 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.

[3] A. Bolotov and M. Fisher. A Clausal Resolution Method for CTL Branching Time Temporal Logic. *Journal of Exper-*

imental and Theoretical Artificial Intelligence., 11:77–93, 1999.

[4] A. Bolotov, M. Fisher, and C. Dixon. On the Relationship between 'w'-automata and Temporal Logic Normal Form. *Journal of Logic and Computation*, 12:561–581, 2002.

[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. M. Clarke and E. A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logic of Programs. Proceedings of Workshop*, volume 131 of Lecture Notes in Computer Science, pages 52–71. Springer, 1981.

[7] 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.

[8] 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.

[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 *STOC 1984, Proceedings of*, 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. Szalas, editors, *Time and Logic, a computational approach*, chapter 3, pages 131–178. UCL Press Limited, 1995.

Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and Fourth International Conference on Temporal Logic (TIME-ICTL'03) 1530-1311/03 $17.00 © 2003 IEEE