Clausal Resolution for CTL*

Alexander Bolotov, Clare Dixon and Michael Fisher

Department of Computing and Mathematics
Manchester Metropolitan University, Manchester M1 5GD, U.K.
{A.Bolotov,C.Dixon,M.Fisher}@doc.mmu.ac.uk

Abstract. In this paper we consider proof techniques for branching-time temporal logics. While a considerable amount of research has been carried out regarding the relationship between finite automata and such logics, practical proof techniques for such logics have received relatively little attention. Recently, however, several applications requiring refined proof methods for branching-time temporal logics have appeared, most notably the specification and verification of multi-agent systems. Thus, here we extend our clausal resolution method for linear-time temporal logics to a branching-time framework, in particular to the powerful CTL* logic. The key elements of the resolution method, namely the normal form, the concept of step resolution and a novel temporal resolution rule, are introduced, justified, and applied.

1 Introduction

A proof method based upon clausal resolution has been developed for *linear* discrete temporal logics [10] and has been shown to be particularly amenable to efficient implementation [6]. It is based upon a normal form that can potentially represent a range of temporal logics, utilising a variety of model structures [11]. For example, in [4] we extended the resolution method to the (comparatively simple) branching time temporal logic CTL [5].

We here consider the extension of this approach to the more powerful branching-time logic CTL* that is now being applied, for example, within the specification and verification of multi-agent systems [15]. The key elements of the method, namely the normal form, the concept of *step* resolution and the form of the *temporal* resolution rule, are introduced and justified with respect to CTL*.

Our approach follows the observation that, as the branching structures characterised by CTL* consist of a set of linear paths, we can use our linear-time temporal resolution along a given path, while using an additional mechanism in order to cope with resolution *between* paths. This is achieved by extending the normal form so that each temporal formula is labelled with an index identifying the path on which it is relevant. Thus, resolution can only be carried out between two formulae if their indices 'match'.

The structure of this paper is as follows. In §2 we briefly outline the syntax and semantics of CTL*. In §3 we define a normal form used for CTL* formulae, introducing the notion of *indices* and providing the context over which the linear-time operators range. In §3.1 the interpretation of such indexed formulae is given. Then in §3.2 we describe the algorithm to transform arbitrary CTL* formulae into the normal form, and in §§3.3-3.6 we present the range of transformation rules. An example of the transformation of a CTL* formula to its normal is given in §3.7. In §4 we introduce a resolution method and the range of resolution rules. In §4.5 we give an example of the resolution refutation. Correctness arguments are outlined in §5. Finally, in §6, we provide concluding remarks and discuss future work.

2 Full Computation Tree Logic — CTL*

The syntax of CTL* distinguishes *state* (S) and *path* (P) formulae. These are defined inductively as follows.

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

Here, C is any well-formed formula of propositional logic, $\mathbf{true}$ and $\mathbf{false}$ are constants, $\mathbf{A}$ ('on all future paths starting here') and $\mathbf{E}$ ('on some future path starting here') are branching-time path operators, and $\Box$ ('always in the future'), $\Diamond$ ('at some time in the future'), $\bigcirc$ ('at the next moment in time'), $\mathcal{U}$ ('until'), and $\mathcal{W}$ ('unless') are linear-time temporal operators. Thus, the very expressive language of CTL* allows us to represent such complex properties as $\mathbf{A}\Diamond(\bigcirc B \wedge \mathbf{E}\bigcirc\neg B)$.

Following [8], we interpret a well-formed formula of CTL* in a tree-like model structure $\mathcal{M} = \langle S, R, L\rangle$, where S is a set of states, $R \subset S \times S$ is a binary relation over S such that there is a state s_0 which is the root of the structure's tree (i.e. $\forall j.\ (\langle s_j, s_0\rangle \notin R)$) and every state has at least one successor (i.e. $\forall i.\ \exists j.\ (\langle s_i, s_j\rangle \in R)$), and L is an interpretation function mapping atomic propositional symbols to truth values at each state.

Before continuing with the semantics of CTL* we first introduce some notation.

A *path*, χ_{s_i}, over R, is a sequence of states $s_i, s_{i+1}, s_{i+2}, \ldots, s_{i+n}, \ldots$ such that $\forall j \geq i.\ (s_j, s_{j+1}) \in R$. 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 an infinite sub-sequence $s_j, s_{j+1}, s_{j+2}, \ldots$ ($s_{j+k} \in \chi_{s_i}$, for $k = 0, 1, \ldots$) a *suffix* of a path χ_{s_i} abbreviating it with $Suf(\chi_{s_i}, s_j)$. We can now give, in Figure 1, the definition of the satisfaction relation '$\models$'.

Definition 1 (Satisfiability). *A well-formed formula, B, is satisfiable if, and only if, there exists a model structure $\mathcal{M}$ such that $\langle \mathcal{M}, s_0\rangle \models B$.*

Definition 2 (Validity). *A well-formed formula, B, is valid if, and only if, B is satisfied in every possible model, i.e. for each model structure $\mathcal{M}$, $\langle \mathcal{M}, s_0\rangle \models B$.*

Note that the CTL* semantics above requires that, when interpreting atomic formulae on some path χ_{s_i}, we are referring to the behaviour in the first state s_i of this path. This causes the *failure of the substitution principle* for proposition symbols and hence induces some restrictions on the *renaming* procedure which plays a significant role in our resolution method. Further, note that in [9] it was shown that any CTL* formula G can be transformed to a particular form G' where the nesting of path quantifiers is at most 2 and where $\models G'$ iff $\models G$. This is achieved by repeatedly renaming the state subformulae of G. We later utilize this property in transforming formulae (§3.2).

$\langle \mathcal{M}, s_i \rangle \models p$	iff	$p \in L(s_i)$, for atomic p
$\langle \mathcal{M}, s_i \rangle \models \neg A$	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 $\chi_{s_i} \langle \mathcal{M}, \chi_{s_i} \rangle \models B$
$\langle \mathcal{M}, s_i \rangle \models \mathbf{E}B$	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
$\langle \mathcal{M}, \chi_{s_i} \rangle \models \neg A$	iff	$\langle \mathcal{M}, \chi_{s_i} \rangle \not\models A$
$\langle \mathcal{M}, \chi_{s_i} \rangle \models A \wedge B$	iff	$\langle \mathcal{M}, \chi_{s_i} \rangle \models A$ and $\langle \mathcal{M}, \chi_{s_i} \rangle \models B$
$\langle \mathcal{M}, \chi_{s_i} \rangle \models A \vee B$	iff	$\langle \mathcal{M}, \chi_{s_i} \rangle \models A$ or $\langle \mathcal{M}, \chi_{s_i} \rangle \models B$
$\langle \mathcal{M}, \chi_{s_i} \rangle \models A \Rightarrow B$	iff	$\langle \mathcal{M}, \chi_{s_i} \rangle \not\models A$ or $\langle \mathcal{M}, \chi_{s_i} \rangle \models B$
$\langle \mathcal{M}, \chi_{s_i} \rangle \models \Box B$	iff	for each $s_j \in \chi_{s_i}$ if $i \leq j$ then $\langle \mathcal{M}, \mathit{Suf}(\chi_{s_i}, s_j) \rangle \models B$
$\langle \mathcal{M}, \chi_{s_i} \rangle \models \Diamond B$	iff	there exists $s_j \in \chi_{s_i}$ such that $i \leq j$ 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 $i \leq j$ and $\langle \mathcal{M}, \mathit{Suf}(\chi_{s_i}, s_j) \rangle \models B$ and for each $s_k \in \chi_{s_i}$ if $i \leq k < j$ 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. CTL* semantics

3 Normal Form for CTL*

The normal form we use for CTL* is called $\mathrm{SNF}_{\mathrm{C}^\star}$. To define $\mathrm{SNF}_{\mathrm{C}^\star}$, we must extend the CTL* language slightly. Firstly, we introduce a new constant $\mathbf{start}$:

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

Secondly, we introduce indices in order to express the path context of a temporal formula. In general, the index of a formula will tell us whether we can reason about this formula in the context of all paths (any arbitrary path) or in the context of some specific path. Indices are based on the two sets $\mathsf{VAR} = \{\alpha, \beta, \gamma, \ldots\}$ representing "path variables" and $\mathsf{FUN} = \{f, g, h, \ldots\}$ representing unary path functions. The set of path expressions, IND, is made up from members of VAR and expressions of the form $f(\alpha)$, where $f \in \mathsf{FUN}$ and $\alpha \in \mathsf{VAR}$. Note that indices of the form $f(g(\alpha))$ can not be obtained as part of our transformation procedure (see §3.8). We will also see later (§3.4) that path expressions of the type $f(\alpha)$ appear due to the removal of the $\mathbf{E}$ quantifier and play a similar role to the Skolem functions in predicate logic.
A formula in $\mathrm{SNF}_{\mathrm{C}^\star}$ is of the form

$$\mathbf{A}\Box \bigwedge_{i=1}^{n} (P_i \Rightarrow F_i)$$

where each "$P_i \Rightarrow F_i$" (called a *rule*) is further restricted to be one of the following:

$$\begin{array}{lll} \mathbf{start} \Rightarrow \bigvee_{k=1}^{l} q_k & & \text{(an } \textit{initial} \text{ rule)} \\ \bigwedge_{j=1}^{m} p_{j\langle\mathsf{var}\rangle} \Rightarrow \bigcirc \bigvee_{k=1}^{l} q_{k\langle\mathsf{ind}\rangle} & & \text{(a } \textit{step} \text{ rule)} \\ \bigwedge_{j=1}^{m} p_{j\langle\mathsf{var}\rangle} \Rightarrow \Diamond l_{\langle\mathsf{ind}\rangle} & & \text{(a } \textit{sometime} \text{ rule)} \end{array}$$

Note that, p_j, q_j and l are literals, $\langle\mathsf{var}\rangle$ and $\langle\mathsf{ind}\rangle$ are indices providing the path *context* for the rules, $\langle\mathsf{var}\rangle \in \mathsf{VAR}$, and $\langle\mathsf{ind}\rangle \in \mathsf{IND}$. Note also that, in an indexed formula such as $C_{\langle\mathsf{ind}_1\rangle}$, the index $\langle\mathsf{ind}_1\rangle$ relates to the whole formula C.

3.1 Interpreting Indexed Formulae

Any $\mathrm{SNF}_{\mathrm{C}^\star}$ rule can be interpreted as a constraint upon the branching structure (remembering that $\mathrm{SNF}_{\mathrm{C}^\star}$ rules are in the scope of an implicit '$\mathbf{A}\Box$'). As a simple example, consider the set of rules $\{\, \mathbf{start} \Rightarrow x,\ x_{\langle\beta\rangle} \Rightarrow \Diamond z_{\langle f(\beta)\rangle},\ \mathbf{true}_{\langle\alpha\rangle} \Rightarrow \bigcirc\neg z_{\langle\alpha\rangle} \,\}$. Indexed formulae of this set are interpreted in relation to some model $\mathcal{M}$ as follows:

- The initial rule $\mathbf{start} \Rightarrow x$ is understood as "x is satisfied at the initial state of $\mathcal{M}$".
- The 'sometime' rule $x_{\langle\beta\rangle} \Rightarrow \Diamond z_{\langle f(\beta)\rangle}$ can be interpreted as "for any fullpath χ and any state $s_i \in \chi$, (indicated by index $\langle\beta\rangle \in \mathsf{VAR}$) if x is satisfied at a state s_i then $\Diamond z$ must be satisfied along some path π_{s_i} (indicated by index $\langle f(\beta)\rangle$)".
- The step rule $\mathbf{true}_{\langle\alpha\rangle} \Rightarrow \bigcirc\neg z_{\langle\alpha\rangle}$ can be interpreted as "for any fullpath χ and any state $s_i \in \chi$, if $\mathbf{true}$ is satisfied at a state s_i then $\bigcirc\neg z$ must be satisfied along any path π_{s_i}".

3.2 Algorithm for Transforming CTL* Formulae to $\mathrm{SNF}_{\mathrm{C}^\star}$

We now consider how an arbitrary CTL* formula can be transformed into $\mathrm{SNF}_{\mathrm{C}^\star}$. Preserving or changing a path index introduced at some stage of the transformation procedure is important in formulating the transformation rules.

To check the validity of some CTL* formula, F, we first negate it and push the negations into F until they are applied to propositions. This is based on negations of classical logic operators in addition to the following equivalences.

$$\begin{array}{lll} \neg\mathbf{A}P \equiv \mathbf{E}\neg P & \neg\mathbf{E}P \equiv \mathbf{A}\neg P & \neg\bigcirc P \equiv \bigcirc\neg P \\ \multicolumn{2}{l}{\neg(P\,\mathcal{U}\,Q) \equiv \neg Q\,\mathcal{W}\,(\neg P \wedge \neg Q)} & \neg\Diamond F \equiv \Box\neg F \\ \multicolumn{2}{l}{\neg(P\,\mathcal{W}\,Q) \equiv \neg Q\,\mathcal{U}\,(\neg P \wedge \neg Q)} & \neg\Box P \equiv \Diamond\neg P \end{array}$$

This gives us a formula G such that $\langle\mathcal{M}, s_0\rangle \models F$ iff $\langle\mathcal{M}, s_0\rangle \models G$. Then the transformation procedure, τ, is applied to G, giving $\tau[G] = \tau_2[\tau_1[G]]$ where τ_1 and τ_2 are described by the steps 1-3 and 4-8 below, respectively.

1. *Anchor* G to $\mathbf{start}$, obtaining $\mathbf{start} \Rightarrow G$.

2. Apply the initial renaming rule obtaining $\mathbf{A}\Box(\mathbf{start} \Rightarrow x_0) \wedge \mathbf{A}\Box(x_{0\langle \mathsf{var}\rangle} \Rightarrow G_{\langle \mathsf{var}\rangle})$, where x_0 is a new proposition.
3. Reduce the nesting of path quantifiers in G gradually renaming the deepest embedded state subformulae (see §3.3).
   This reduction of the nesting of state formulae in G is based on the method defined in [9]. Once this has been carried out, we obtain a set of constraints of the form $\mathbf{A}\Box(B_{\langle \mathsf{var}\rangle} \Rightarrow \mathsf{P}C_{\langle \mathsf{var}\rangle})$ or $\mathbf{A}\Box(B_{\langle \mathsf{var}\rangle} \Rightarrow C_{\langle \mathsf{var}\rangle})$ where P is either of the path operators and C a formula without path operators.
4. Remove path quantifiers (see §3.4); this provides a context for the renaming of path formulae in the next step.
5. Rename path formulae (see §3.3), preserving the path context, in order to reduce the nesting of temporal operators to exactly 1 and so that every temporal operator applies only to literals.
6. Remove temporal operators (see §3.5).
7. Introduce a temporal context (see §3.6), *where necessary*, use classical equivalences to rewrite rules into their correct form.
8. Rename path variables apart, again if necessary, to ensure that no path variable occurs in two different rules.

3.3 Renaming rule

We extract subformulae from within a complex formula as follows.

$$P_{\langle \mathsf{ind}_1\rangle} \Rightarrow Q(R)_{\langle \mathsf{ind}_2\rangle} \longrightarrow \{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow Q(R/x)_{\langle \mathsf{ind}_2\rangle},\ \ x_{\langle \mathsf{ind}_3\rangle} \Rightarrow R_{\langle \mathsf{ind}_2\rangle}\}$$

Here, $Q(R)$ means "R is a subformula of (Q)", $Q(R/x)$ means a result of replacing R by a new proposition symbol x in Q and $\langle \mathsf{ind}_3\rangle = \langle \mathsf{ind}_2\rangle$ if $\langle \mathsf{ind}_2\rangle \in \mathsf{VAR}$, but if $\langle \mathsf{ind}_2\rangle = \langle f(\alpha)\rangle$ then $\langle \mathsf{ind}_3\rangle = \langle \alpha\rangle$.

3.4 Removal of Path Quantifiers

These rules introduce new indices representing a path context of the appropriate type.

Removal of A:

$$\frac{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow \mathbf{A}F_{\langle \mathsf{ind}_2\rangle}}{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow F_{\langle \mathsf{ind}_3\rangle}} \quad (\mathsf{ind}_3 \in \mathsf{VAR})$$

Removal of E:

$$\frac{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow \mathbf{E}F_{\langle \mathsf{ind}_2\rangle}}{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow F_{\langle f(\mathsf{ind}_2)\rangle}} \quad (\mathsf{f} \in \mathsf{FUN})$$

3.5 Removal of Temporal Operators

Here we must correctly maintain a path context while reducing temporal operators using their fixed point definitions. In the formulation of the removal rules below 'x' is a new proposition symbol, $\langle \mathsf{ind}_3\rangle \in \mathsf{VAR}$, and $\langle \mathsf{ind}_3\rangle = \langle \mathsf{ind}_2\rangle$ if $\langle \mathsf{ind}_2\rangle \in \mathsf{VAR}$ but if $\langle \mathsf{ind}_2\rangle = \langle f(\alpha)\rangle$ then $\langle \mathsf{ind}_3\rangle = \langle \alpha\rangle$.

Removal of 'always':

$$\frac{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow \Box F_{\langle \mathsf{ind}_2\rangle}}{\begin{array}{l} P_{\langle \mathsf{ind}_1\rangle} \Rightarrow (F \wedge x)_{\langle \mathsf{ind}_1\rangle} \\ x_{\langle \mathsf{ind}_3\rangle} \Rightarrow \bigcirc(F \wedge x)_{\langle \mathsf{ind}_2\rangle} \end{array}}$$

Removal of 'unless':

$$\frac{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow (F\,\mathcal{W}\,G)_{\langle \mathsf{ind}_2\rangle}}{\begin{array}{l} P_{\langle \mathsf{ind}_1\rangle} \Rightarrow (G \vee (F \wedge x))_{\langle \mathsf{ind}_1\rangle} \\ x_{\langle \mathsf{ind}_3\rangle} \Rightarrow \bigcirc(G \vee (F \wedge x))_{\langle \mathsf{ind}_2\rangle} \end{array}}$$

Removal of 'until':

$$\frac{P_{\langle \mathsf{ind}_1\rangle} \Rightarrow (F\,\mathcal{U}\,G)_{\langle \mathsf{ind}_2\rangle}}{\begin{array}{l} P_{\langle \mathsf{ind}_1\rangle} \Rightarrow (G \vee (F \wedge x))_{\langle \mathsf{ind}_1\rangle} \\ x_{\langle \mathsf{ind}_3\rangle} \Rightarrow \bigcirc(G \vee (F \wedge x))_{\langle \mathsf{ind}_2\rangle} \\ P_{\langle \mathsf{ind}_1\rangle} \Rightarrow \Diamond G_{\langle \mathsf{ind}_2\rangle} \end{array}}$$

3.6 Introducing a Temporal Context and Simplifying

Since, for a purely classical formula, 'F', it is the state, rather than the current path, that is important in establishing satisfiability, we can apply the following rules (where F is classical and 'P' is either of the path quantifiers).

Temporising:
$$\frac{Q_{\langle \mathsf{ind}_1 \rangle} \Rightarrow F_{\langle \mathsf{ind}_2 \rangle}}{\begin{array}{rl} \mathbf{start} & \Rightarrow (\neg Q \vee F) \\ \mathbf{true}_{\langle \mathsf{ind}_1 \rangle} & \Rightarrow \bigcirc(\neg Q \vee F)_{\langle \mathsf{ind}_1 \rangle} \end{array}}$$

Further, we use a number of transformations that correspond to the following equivalences of CTL*: $\bigcirc(P \wedge Q) \equiv \bigcirc P \wedge \bigcirc Q$, $\mathbf{EA}F \equiv \mathbf{A}F$; $\mathbf{AE}F \equiv \mathbf{E}F$, $\mathbf{AA}F \equiv \mathbf{A}F, \mathbf{EE}F \equiv \mathbf{E}F$ and obvious simplifications $\mathsf{T}\mathbf{false} \equiv \mathbf{false}$, $\mathsf{PT}\mathbf{false} \equiv \mathbf{false}$, where 'P' is either of the path quantifiers and 'T' is any unary temporal operator. Finally, we utilize transformations applied to obtain normal form in classical logic; we term this set of additional simplifications SIMP and apply them wherever required.

3.7 Example Transformation

Let us consider the steps required to transform the formula $\mathbf{A}\diamondsuit(\bigcirc p \wedge \mathbf{E}\bigcirc\neg p)$ (whose unsatisfiability is not immediately obvious) into $\mathrm{SNF}_{\mathrm{C}^\star}$.

Recall that, following the transformation algorithm, we first apply the initial renaming rule (steps 2 and 3 below) and then rename the deepest embedded state subformula (steps 4 and 5).

1.	$\mathbf{start} \Rightarrow \mathbf{A}\diamondsuit(\bigcirc p \wedge \mathbf{E}\bigcirc\neg p)$	*Given*
2.	$\mathbf{start} \Rightarrow x$	1, Initial Renaming
3.	$x_{\langle\alpha\rangle} \Rightarrow \mathbf{A}\diamondsuit(\bigcirc p \wedge \mathbf{E}\bigcirc\neg p)_{\langle\alpha\rangle}$	1, Initial Renaming
4.	$x_{\langle\alpha\rangle} \Rightarrow \mathbf{A}\diamondsuit(\bigcirc p \wedge y)_{\langle\alpha\rangle}$	3, Renaming
5.	$y_{\langle\alpha\rangle} \Rightarrow \mathbf{E}\bigcirc\neg p_{\langle\alpha\rangle}$	3, Renaming

As the nesting of path quantifiers is now of depth 1 (in rules 4 and 5), we remove these quantifiers (steps 6 and 7) thus generating a certain path context for purely path formulae.

6.	$x_{\langle\alpha\rangle} \Rightarrow \diamondsuit(\bigcirc p \wedge y)_{\langle\delta\rangle}$	4, **A** Removal
7.	$y_{\langle\alpha\rangle} \Rightarrow \bigcirc\neg p_{\langle f(\alpha)\rangle}$	5, **E** Removal

Now we can rename a purely path formula $(\bigcirc p \wedge y)$ which occurs in the context $\diamondsuit(\bigcirc p \wedge y)_{\langle\delta\rangle}$ (steps 8 and 9) and then apply simplification and temporising rules.

8.	$x_{\langle\alpha\rangle} \Rightarrow \diamondsuit z_{\langle\delta\rangle}$	6, Renaming
9.	$z_{\langle\delta\rangle} \Rightarrow (\bigcirc p \wedge y)_{\langle\delta\rangle}$	6, Renaming
10.	$z_{\langle\delta\rangle} \Rightarrow \bigcirc p_{\langle\delta\rangle}$	9, SIMP
11.	$z_{\langle\delta\rangle} \Rightarrow y_{\langle\delta\rangle}$	9, SIMP
12.	$\mathbf{start} \Rightarrow (\neg z \vee y)$	11, Temporising
13.	$\mathbf{true}_{\langle\delta\rangle} \Rightarrow \bigcirc(\neg z \vee y)_{\langle\delta\rangle}$	11, Temporising

Finally, taking 2, 7, 8, 10, 12 and 13 above and renaming path variables (in 7, 10 and 13) apart we obtain the desired set of $\mathrm{SNF}_{\mathrm{C}^\star}$ rules. (In §4.5 we present a resolution proof for this set of rules, repeating it as a part of such proof).

3.8 Features of $\mathrm{SNF}_{\mathrm{C}^\star}$

Here we summarize some important features of the transformation procedure that can be proved straightforwardly from the transformation algorithm given above.

1. Labels of the type $\langle f(\mathsf{ind})\rangle$ can appear only on the right hand side of a rule and therefore any index on the left hand side is always a path variable.
2. If the index of the right hand side formula is $\langle f(\mathsf{ind})\rangle$ then the left hand side is always labeled by $\langle \mathsf{ind}\rangle$ which ensures the link between left and right hand side indices.
3. We can not obtain indices where an argument of a function is itself a function, i.e. of the type $\langle f(g(\alpha))\rangle$. Note however, that use of the indices of the type $\langle f(\mathsf{ind})\rangle$ is crucial as a function symbol f is a syntactical indication of a certain path context (see §3.1).
4. On the left hand side of formulae that appear during the transformation procedure we can only have an expression which is either of the following types: $\mathbf{start}$, $\mathbf{true}$, a literal or a conjunction of literals.

3.9 Correctness of the transformation

The following theorem (see [3] for details) characterizes the correctness of the transformation procedure τ.

Theorem 1. *A well-formed* CTL* *formula, G, is satisfiable if, and only if,* $\tau(G)$ *is.*

4 Resolution procedure for CTL*

Once the original formula has been transformed into $\mathrm{SNF}_{\mathrm{C}^\star}$, there are two possible situations in which resolution rules can be applied. The first type of resolution rule, which we call *step resolution*, (see §4.1), is applied when literals l and $\neg l$ occur at the same moment of time on the same branch. The second type of resolution rule, called *temporal resolution* (see §4.2), can be applied when some proposition l can occur at all future moments on a path (a situation known as a *loop*), while l can be also constrained to be false at some point in the future of the same path.

In both cases, the use of indices is crucial as they express the required path contexts. As we will see below, we must be able to unify indices in order to carry out resolution. Unification between indices is the same as the unification of terms in classical logic.

4.1 Step Resolution

With l as a literal and $\theta(\langle \mathsf{ind_a}\rangle, \langle \mathsf{ind_b}\rangle)$ as an abbreviation for the unification of two indices, we have the following step resolution rules.

SRES1:

$$\frac{\begin{array}{r} P_{\langle \mathsf{ind}_1\rangle} \Rightarrow \bigcirc (C \vee l)_{\langle \mathsf{ind}_2\rangle} \\ Q_{\langle \mathsf{ind}_3\rangle} \Rightarrow \bigcirc (D \vee \neg l)_{\langle \mathsf{ind}_4\rangle} \end{array}}{(P \wedge Q)_{\langle \mathsf{ind}_5\rangle} \Rightarrow \bigcirc (C \vee D)_{\langle \mathsf{ind}_6\rangle}}$$

SRES2:

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

where $\langle \mathsf{ind}_6\rangle = \theta(\langle \mathsf{ind}_2\rangle, \langle \mathsf{ind}_4\rangle)$ and, if $\langle \mathsf{ind}_6\rangle \in \mathsf{VAR}$ then $\langle \mathsf{ind}_5\rangle = \langle \mathsf{ind}_6\rangle$ but if $\langle \mathsf{ind}_6\rangle$ is of the form $\langle f(\alpha)\rangle$ then $\langle \mathsf{ind}_5\rangle = \langle \alpha\rangle$.

4.2 Temporal Resolution

To apply the temporal resolution rule (defined below) we must first consider a *loop*. A *loop* is a situation when a literal, say l, occurs at all future moments on some or all paths. In the linear-time case an algorithm for identifying loops in sets of merged rules has been developed [6] and we expect to extend this technique to the case of CTL*. Merged rules are generated from step rules as follows.

$$\frac{\begin{array}{c} R_{\langle \mathsf{ind}_1 \rangle} \Rightarrow \bigcirc C_{\langle \mathsf{ind}_2 \rangle} \\ Q_{\langle \mathsf{ind}_3 \rangle} \Rightarrow \bigcirc D_{\langle \mathsf{ind}_4 \rangle} \end{array}}{(R \wedge Q)_{\langle \mathsf{ind}_5 \rangle} \Rightarrow (C \wedge D)_{\langle \mathsf{ind}_6 \rangle}}$$

where $\langle \mathsf{ind}_6 \rangle = \theta(\langle \mathsf{ind}_2 \rangle, \langle \mathsf{ind}_4 \rangle)$ and if $\langle \mathsf{ind}_6 \rangle \in \mathsf{VAR}$ then $\langle \mathsf{ind}_5 \rangle = \langle \mathsf{ind}_6 \rangle$ else if $\langle \mathsf{ind}_6 \rangle$ is of the form $f(\alpha)$ then $\langle \mathsf{ind}_5 \rangle = \langle \alpha \rangle$.

Definition 3 (Loop in CTL***).** *A loop in l is a set of merged rules*

$$\left\{ \begin{array}{c} P_{0_{\langle \mathsf{ind}_{0_1} \rangle}} \Rightarrow \bigcirc Q_{0_{\langle \mathsf{ind}_{0_2} \rangle}} \\ \ldots \quad \ldots \\ P_{n_{\langle \mathsf{ind}_{n_1} \rangle}} \Rightarrow \bigcirc Q_{n_{\langle \mathsf{ind}_{n_2} \rangle}} \end{array} \right\}$$

such that for all $0 \leq i \leq n$, *both* $\models Q_i \Rightarrow l$ *and* $\models Q_i \Rightarrow \bigvee_{j=0}^{n} P_j$.

We will abbreviate such loop by $(P_0 \vee \ldots \vee P_n)_{\langle \mathsf{ind}_1 \rangle} \Rightarrow \bigcirc \Box l_{\langle \mathsf{ind}_2 \rangle}$ where, if for all i, $\langle \mathsf{ind}_{i_2} \rangle \in \mathsf{VAR}$ then $\langle \mathsf{ind}_1 \rangle = \langle \mathsf{ind}_2 \rangle = \langle \alpha \rangle$ where $\langle \alpha \rangle$ is a new index in VAR, but if for all i $\langle \mathsf{ind}_{i_2} \rangle$ only involves one function symbol, say f, then $\langle \mathsf{ind}_2 \rangle = \langle f(\alpha) \rangle$ and $\langle \mathsf{ind}_1 \rangle = \langle \alpha \rangle$ where $\langle \alpha \rangle$ is a new index in VAR, else $\langle \mathsf{ind}_2 \rangle = \langle h(\alpha) \rangle$ and $\langle \mathsf{ind}_1 \rangle = \langle \alpha \rangle$ where $\langle \alpha \rangle$ is a new index in VAR and h is a new function symbol in FUN.

For this set of merged rules, each right hand side implies one or more left hand sides from the side condition on loops. As indices on the right hand sides are either functions or variables, and indices on the left hand sides are always variables, right hand side indices will always unify with the relevant left hand side indices. Each right hand side implies l. Hence, once one of the left hand sides is satisfied a literal l holds at all future moments on some or all paths (dependent on the type of the index).

Now, once we have detected $(P_0 \vee \ldots \vee P_n)_{\langle \mathsf{ind}_1 \rangle} \Rightarrow \bigcirc \Box l_{\langle \mathsf{ind}_2 \rangle}$ indicating a loop in l we can resolve it with a sometime rule $Q_{\langle \mathsf{ind}_3 \rangle} \Rightarrow \Diamond \neg l_{\langle \mathsf{ind}_4 \rangle}$ provided the indices $\langle \mathsf{ind}_2 \rangle$ and $\langle \mathsf{ind}_4 \rangle$ can be unified.

$$\textbf{TRES:} \quad \frac{\begin{array}{l} P_{\langle \mathsf{ind}_1 \rangle} \Rightarrow \bigcirc \Box l_{\langle \mathsf{ind}_2 \rangle} \\ Q_{\langle \mathsf{ind}_3 \rangle} \Rightarrow \Diamond \neg l_{\langle \mathsf{ind}_4 \rangle} \end{array}}{Q_{\langle \mathsf{ind}_5 \rangle} \Rightarrow (\neg P \, \mathcal{W} \, \neg l)_{\langle \mathsf{ind}_4 \rangle}}$$

where $\theta(\langle \mathsf{ind}_2 \rangle, \langle \mathsf{ind}_4 \rangle) \neq \emptyset$ and $\langle \mathsf{ind}_5 \rangle = \langle \mathsf{ind}_4 \rangle$ if $\langle \mathsf{ind}_4 \rangle \in \mathsf{VAR}$ else if $\langle \mathsf{ind}_4 \rangle$ is of the form $f(\alpha)$ then $\langle \mathsf{ind}_5 \rangle = \langle \alpha \rangle$.

Note that in the special case of TRES, when $\Diamond \neg l$ is labeled by $\langle \mathsf{ind}_4 \rangle \in \mathsf{VAR}$, we preserve this label for the conclusion even when $\langle \mathsf{ind}_2 \rangle$ is $\langle f(\alpha) \rangle$. This is related to the limit closure property of CTL* ([8]) and is required for completeness ([3]).

Observe also that the conclusion of TRES should be further translated into $\mathrm{SNF}_{\mathrm{C}^\star}$.

4.3 Transferring Constraints

If at any point we derive a formula such as $(P \wedge \ldots \wedge Q)_{\langle \mathsf{ind}_1 \rangle} \Rightarrow \bigcirc \mathbf{false}_{\langle \mathsf{ind}_2 \rangle}$ then we must ensure that $P \wedge \ldots \wedge Q$ never occurs anywhere by applying the following rule.

Transferral rule:
$$\frac{(P \wedge \ldots \wedge Q)_{\langle \mathsf{ind}_1 \rangle} \Rightarrow \bigcirc \mathbf{false}_{\langle \mathsf{ind}_2 \rangle}}{\begin{array}{rl} \mathbf{start} & \Rightarrow (\neg P \vee \ldots \neg Q) \\ \mathbf{true}_{\langle \mathsf{ind}_1 \rangle} & \Rightarrow \bigcirc (\neg P \vee \ldots \vee \neg Q)_{\langle \mathsf{ind}_1 \rangle} \end{array}}$$

4.4 Termination

The step and temporal resolution rules are repeatedly applied. If we reach a stage where no new resolvents are generated, then the procedure terminates. If either $\mathbf{start} \Rightarrow \mathbf{false}$ or $\mathbf{true}_{\langle \mathsf{ind}_1 \rangle} \Rightarrow \bigcirc \mathbf{false}_{\langle \mathsf{ind}_2 \rangle}$ are generated during the temporal resolution procedure, it terminates and the original set of rules is unsatisfiable (see §5).

4.5 Resolution Example

Now we consider a resolution refutation for $\mathbf{A}\Diamond(\bigcirc p \wedge \mathbf{E}\bigcirc \neg p)$ given as a set of $\mathrm{SNF}_{\mathrm{C}^\star}$ rules obtained in §3.7. We begin the proof repeating this set of rules.

1.	$\mathbf{start}$	$\Rightarrow x$	$\mathrm{SNF}_{\mathrm{C}^\star}$
2.	$\mathbf{start}$	$\Rightarrow \neg z \vee y$	$\mathrm{SNF}_{\mathrm{C}^\star}$
3.	$\mathbf{true}_{\langle \epsilon \rangle}$	$\Rightarrow \bigcirc(\neg z \vee y)_{\langle \epsilon \rangle}$	$\mathrm{SNF}_{\mathrm{C}^\star}$
4.	$y_{\langle \beta \rangle}$	$\Rightarrow \bigcirc \neg p_{\langle f(\beta) \rangle}$	$\mathrm{SNF}_{\mathrm{C}^\star}$
5.	$z_{\langle \varphi \rangle}$	$\Rightarrow \bigcirc p_{\langle \varphi \rangle}$	$\mathrm{SNF}_{\mathrm{C}^\star}$
6.	$x_{\langle \alpha \rangle}$	$\Rightarrow \Diamond z_{\langle \delta \rangle}$	$\mathrm{SNF}_{\mathrm{C}^\star}$
7.	$(y \wedge z)_{\langle \beta \rangle}$	$\Rightarrow \bigcirc \mathbf{false}_{\langle f(\beta) \rangle}$	$4, 5,$ SRES1
8.	$\mathbf{start}$	$\Rightarrow \neg y \vee \neg z$	$7,$ Transferral
9.	$\mathbf{true}_{\langle \beta \rangle}$	$\Rightarrow \bigcirc(\neg y \vee \neg z)_{\langle \beta \rangle}$	$7,$ Transferral
10.	$\mathbf{true}_{\langle \beta \rangle}$	$\Rightarrow \bigcirc \neg z_{\langle \beta \rangle}$	$3, 9,$ SRES1
11.	$x_{\langle \delta \rangle}$	$\Rightarrow (\mathbf{false}\, \mathcal{W}\, z)_{\langle \delta \rangle}$	$6, 10,$ TRES
12.	$x_{\langle \delta \rangle}$	$\Rightarrow (z \vee (\mathbf{false} \wedge r))_{\langle \delta \rangle}$	$11,$ Removal of $\mathcal{W}$
13.	$x_{\langle \delta \rangle}$	$\Rightarrow z_{\langle \delta \rangle}$	$12,$ SIMP
14.	$\mathbf{start}$	$\Rightarrow \neg x \vee z$	$13,$ Temporising
15.	$\mathbf{start}$	$\Rightarrow \mathbf{false}$	$1, 14, 2, 8,$ SRES2

5 Correctness of the resolution procedure

Our correctness argument is given by a number of theorems stating fundamental properties of the above resolution procedure.

Theorem 2 (Termination). *Given any set R of* $\mathrm{SNF}_{\mathrm{C}^\star}$ *rules the resolution procedure applied to R terminates.*

Theorem 3 (Soundness of the resolution procedure). *Given a set R of* $\mathrm{SNF}_{\mathrm{C}^\star}$ *rules if there is a resolution refutation for R (see §4.4) then R is unsatisfiable.*

Theorem 4 (Completeness of the resolution procedure). *If a set R of* $\mathrm{SNF}_{\mathrm{C}^\star}$ *rules is unsatisfiable then there exists a resolution refutation for R.*

While the proof of theorem 2, based on the finiteness of R, is relatively simple and the reader is referred to [3], proofs of theorems 3 and 4 are more complex requiring a number of technical definitions from the graph and automata theory. Due to the lack of the space we sketch outlines of these proofs, again referring to [3] for full details.

First of all note, that within the proofs of the theorems 3 and 4, we utilise ideas from the alternating automata approach to temporal logic [13, 14, 2, 17] and essentially use the finite tree model property of CTL*.

The following observation is important for the proofs. The structure of a set of $\mathrm{SNF}_{\mathrm{C}^\star}$ rules (see §3) can be related to that of a transition system. The initial rules provide the starting conditions while the step and eventuality rules can be considered as "global" transition rules. Thus, for a set R of $\mathrm{SNF}_{\mathrm{C}^\star}$ rules obtained for some CTL* formula G we define an infinite *alternating computation tree* which is a type of labeled *AND-OR tree*. Subformulae of R are used as the labels for the states of such tree. Transitions in the AND-OR tree are hyper-transitions allowing for a state t_i to have a set of successors $t_j \dots t_k$ called AND-successors of t_i indicating the application of several rules simultaneously. Hyper-transitions are determined by a transition function defined similarly to one for the hesitant alternating tree automaton ([17]) and using the set of the global rules of R.

Then we show that due to the finite model property of CTL* an infinite computation tree collapses into a finite graph. This graph corresponds to a set of possible runs of non-deterministic Büchi Tree Automaton [9].

A set of the standard deletion rules is applied to the graph: nodes with outstanding eventualities and nodes without successors must be deleted (as paths through these nodes cannot form part of any model of R). This pruning procedure terminates resulting in a Büchi Tree Automaton, $\mathcal{B}_R$, that accepts exactly those trees that satisfy the set R. This proves that a set R of $\mathrm{SNF}_{\mathrm{C}^\star}$ rules is unsatisfiable if and only if the automaton $\mathcal{B}_R$ is empty.

Finally, we show that the deletions in the graph of $\mathcal{B}_R$ correspond to the resolution rules we have developed, thus, proving the fact that a set R of $\mathrm{SNF}_{\mathrm{C}^\star}$ rules is unsatisfiable if and only if there exists a resolution refutation for R.

Therefore, taking into account theorem 1, we conclude that given a CTL* formula G, then if there exists a resolution refutation for the set of $\mathrm{SNF}_{\mathrm{C}^\star}$ rules R generated from G then R (and hence G) is unsatisfiable. If the resolution for R terminates not finding a refutation it is possible to extract a model for R and thus R (and hence G) is satisfiable.

6 Conclusions

We have extended the clausal resolution method developed for linear-time temporal logics to a branching-time framework. This will form the basis of future work into both the efficient implementation of this approach, where we expect to utilise techniques developed for implementing linear-time temporal resolution, and a detailed complexity analysis. In addition, we will also further examine the relationship between the different types of branching-time temporal logics by studying their translation into our normal form.

The authors would like to thank the anonymous referees for useful comments. This work was supported by EPSRC under research grant GR/L87491.

References

1. O. Bernholtz and O. Grumberg. Branching-Time Temporal Logic and Amorphous Tree Automata. In *Proc.of 4th Conference on Concurrency Theory*, Springer-Verlag, (LNCS 715), 1993.
2. O. Bernholtz, M. Vardi and P. Wolper. An Automata-Theoretic Approach to Branching-Time Model Checking. In *Proc. Conf. on Computer-Aided verification (CAV'94)*, June 1994, Springer-Verlag, (LNCS 818), pp 142-155.
3. A. Bolotov. Clausal Resolution for Branching-Time Temporal Logic. PhD Thesis. In preparation.
4. A. Bolotov and M. Fisher A clausal resolution method for CTL branching time temporal Logic. *Journal of experimental and theoretical artificial intelligence*, Taylor & Francis, 11, 1999, pp 77-93.
5. E. M. Clarke and E. A. Emerson. Design and Synthesis of Synchronisation Skeletons Using Branching Time Temporal Logic. Springer-Verlag, (LNCS 131), 1981.
6. C. Dixon. Search Strategies for Resolution in Temporal Logics. In *Proceedings of the Thirteenth International Conference on Automated Deduction (CADE)*, Springer-Verlag, August 1996.
7. E. A. Emerson. Alternative Semantics for Temporal Logics. *Theoretical Computer Science*, 26, 1983, pp 120–130.
8. E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pp 996–1072. Elsevier, 1990.
9. E. A. Emerson. Automated Reasoning about Reactive Systems. In *Logics for Concurrency: Structures Versus Automata*, Springer-Verlag, (LNCS 1043), 1996.
10. M. Fisher. A Resolution Method for Temporal Logic. In *Proc. Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia, 1991. Morgan Kaufman.
11. M. Fisher. A Normal Form for Temporal Logic and its Application in Theorem-Proving and Execution. *Journal of Logic and Computation*, 7(4), August 1997.
12. R. Kaivola. Axiomatizing Extended Computation Tree Logic. In *Proceedings of 21st International Colloquium on Trees in Algebra and Programming – CAAP'96*, volume 1059 of *Lecture Notes in Computer Science*, pp 87–101. Springer-Verlag, 1996.
13. D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. In *Theoretical Computer Science*, pp 267-276, vol.54, 1987.
14. D.E. Muller and P.E. Schupp. Simulating Alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton. In *Theoretical Computer Science*, pp 69-107, vol. 141, 1995.
15. A. S. Rao. Decision procedures for propositional linear-time Belief-Desire-Intention logics. In *Intelligent Agents II (LNAI 1037)*. Springer-Verlag: Heidelberg, Germany, 1996.
16. C. Stirling. Modal and Temporal Logics. In *Handbook of Logic in Computer Science*. Oxford University Press, 1992.
17. M. Y. Vardi. An Automata-theoretic Approach to Linear Temporal Logic. In *Logics for Concurrency: Structures Versus Automata*, Springer-Verlag (LNCS 1043), 1996.