A Resolution Method for CTL Branching-Time Temporal Logic

Alexander Bolotov and Michael Fisher

Department of Computing
Manchester Metropolitan University
Manchester M1 5GD, UK.
{A.Bolotov,M.Fisher}@doc.mmu.ac.uk

Abstract

In this paper we extend our clausal resolution method for linear temporal logics to a branching-time *framework. The branching-time temporal logics considered are* Computation Tree Logic (CTL), *often regarded as the simplest useful logic of this class, and* Extended CTL (ECTL), *which is CTL extended with fairness operators. The key elements of the resolution method, namely the normal form, the concept of* step *resolution and a novel* temporal *resolution rule, are introduced and justified with respect to both these logics. A completeness argument is provided, together with an example of the use of the temporal resolution method. Finally, we consider future work, in particular extension of the method yet further, to CTL*, and implementation of the approach by utilising techniques developed for linear-time temporal resolution.*

1 Introduction

Temporal logic, which was originally developed as a logical framework in which to describe tense in natural languages [13], is now recognized as an essential tool for reasoning about programs. The sequence of states of a program's execution can be considered as a sequence of *moments*, or *states*, within a temporal logic. Thus, proofs about the correctness of programs correspond to proofs within an appropriate temporal logic [5]. A particular area in which temporal logics have been extensively used is in the specification and verification of properties of concurrent and distributed systems [12]. The power of the temporal language allows the representation of a variety of complex properties relating to these systems, such as liveness, deadlock and invariance.

As the applications that require concurrent and distributed solutions have become more refined, so the corresponding logical tools have been extended. In representing the behaviour of concurrent systems, the ability to refer to a range of possible execution paths is seen as important. Thus, there is a need for methods incorporating *branching-time* temporal logics [1]. Here, the underlying model of time is of a choice of possibilities branching into the future. Such branching-time temporal logics have been developed and applied to the specification of concurrent and distributed systems [2].

It has been observed that most correctness properties of concurrent programs (that do not deal with fairness) can be expressed in a branching-time logic called Computation Tree Logic (CTL), first proposed in [1]. There are several extensions of CTL of which CTL* is commonly considered as a full branching-time logic [9]. However, the core logics we concentrate on, are CTL and its extension Extended CTL (ECTL) [5], which incorporates simple fairness constraints. It has been shown that CTL and ECTL can be respectively extended to CTL^+ and $ECTL^+$, where boolean combinations of temporal modalities are allowed. Here, CTl^+ is of equivalent expressive power to CTL [6], however, $ECTL^+$ is strictly more expressive than ECTL.

Much of the research into the verification of concurrent and distributed systems has centred around the *model-checking* technique utilising CTL. Here the satisfiability of a CTL formula is checked with respect to a model derived from a finite-state program [2], [8]. Due to the success of this approach, together with a lack of direct applications of proof in branching-time temporal logics, relatively little research has been carried out on efficient decision procedures for such logics. The work that has been produced has mainly been concerned with basic tableau and automata methods for these logics [6]. However, in recent years several applications of branching-time temporal logics requiring improved proof methods have been developed, most notably the specification and verification of multi-agent systems [14]. This has led to the requirement for more refined, and potentially more efficient, proof methods.

For *linear* discrete temporal logics we have developed a proof method based upon clausal temporal resolution [10]. Due to its simple, yet flexible, formulation, this approach has been shown to be particularly amenable to efficient implementation [4]. It is based upon a normal form that can

0-8186-7937-9/97 $10.00

potentially represent a range of temporal logics, utilising a variety of model structures [11]. We here consider the extension of this approach to CTL and ECTL. The key elements of the method, namely the normal form, the concept of *step* resolution and the form of *temporal* resolution rule, are introduced and justified with respect to both logics.

We first briefly outline the syntax and semantics of CTL (§2). This is a rich temporal language, consisting of the future-time temporal modalities representing "always", "sometimes", "next", "until" and "unless". As our resolution approach is clausal, we next introduce a normal form used for CTL formulae (§3). A CTL formula in normal form is given as a set of formulae of a special structure, where all temporal operators, except "next" and "sometime", have been removed. Such sets can express all information contained in the original formula. If this information concerns a *loop*, i.e., a situation where some proposition Q must occur at all future moments, and if Q is also constrained to be false at some point in the future, then the temporal resolution rule can be applied. The range of resolution and simplification rules used are described in §4 as are simple examples and correctness arguments. In §5, we extend the approach to ECTL, while, in §6, we provide concluding remarks and discuss future work. Finally, in Appendix A, we provide a complete axiomatic system for CTL which is used as part of the completeness proof in §4.

2 Computation Tree Logic (CTL)

2.1 Syntax of CTL

We define the language of Computational Tree Logic (CTL) using the following symbols.

- atomic propositions:
  $p, q, r, \ldots, p_1, q_1, r_1, \ldots, p_n, q_n, r_n, \ldots$
- classical operators: $\neg, \wedge, \Rightarrow, \equiv, \vee,$
- classically defined constants **true**, **false**
- new constant **start**, meaning 'at the beginning of time'
- temporal operators:
  - $\Box$ – 'always in the future'
  - $\Diamond$ – 'at some time in the future'
  - $\bigcirc$ – 'at the next moment in time'
  - $\mathcal{U}$ – 'until'
  - $\mathcal{W}$ – 'unless'
- branching-time path operators:
  - **A** – 'on all future paths starting here'
  - **E** – 'on some future path starting here'

The set of *well-formed formulae* of CTL, WFF_{CTL}, is defined as follows.

1. All atomic propositions, **true**, **false** and **start** are in WFF_{CTL}.
2. If A and B are in WFF_{CTL}, then so are $A \wedge B$, $\neg A$, $A \vee B$, $A \Rightarrow B$, $A \boxminus B$.
3. If A and B are in WFF_{CTL}, then $\Box A$, $\Diamond A$, $\bigcirc A$, $A \,\mathcal{U}\, B$ and $A \,\mathcal{W}\, B$ are all *path* formulae.
4. If P is a path formula then $\mathbf{A}P$ and $\mathbf{E}P$ are both in WFF_{CTL}.

Each CTL formula has a structure where any temporal operator can only be followed by a path operator or a classical operator, while any path operator can only be followed by a temporal operator. As a result CTL is weaker than linear-time temporal logic in its expressive capabilities within a path, but is more expressive in that it can quantify over paths themselves. As an example of its restricted nature, note that no formula describing the property where both $\Diamond A$ and $\Box B$ are satisfied on the same specific path can be constructed using CTL syntax. (Note that there are a variety of natural extensions of CTL, culminating in CTL*, of which some intermediate extensions such as Extended CTL — see §5 — are often useful.)

2.2 Semantics for CTL

Observe that CTL has the following features that are important for our analysis:

- Each state can have an infinite number of successors, but should have at least one.
- Each state belongs to some path (a sequence of states with finite past and infinite future).
- No merging of paths is allowed — a path, once started from another one, has no more common states with any other path.

For the detailed description of CTL's theoretical properties, subsystems and extensions see [5].

A formula of CTL is interpreted in a 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 for all i there exists a j such that $\langle s_i, s_j \rangle \in R$ (as each path is isomorphic to the Natural Numbers, we will abbreviate this relation by '$\leq$') and
- L is an interpretation for all atomic propositional symbols at each state.

In Figure 1 we define a relation '$\models$', (slightly changing the similar definition in [5]) which evaluates well-formed CTL formulae at a particular state s_i in a particular model $\mathcal{M}$.

Notation

1. Any logical constant with '$\bullet$' on top means a metalanguage expression, i.e. $\dot{\exists}$ and $\dot{\Rightarrow}$ mean respectively metalanguage 'some path' and 'implies' (to differentiate them from their language analogues.
2. An infinite path, x_{s_i} $(0 \leq i)$, is a sequence $s_i, s_{i+1}, s_{i+2}, \ldots$ such that $\dot{\forall}\, j\ (\langle s_j, s_{j+1}\rangle \in R)$. x_{s_0} is a path originating from **start**.
3. A state exists that is the root of the structure's tree. We term this s_0 and it satisfies $\dot{\neg}\ \dot{\exists}\, j\ (\langle s_j, s_0\rangle \in R)$.

$$
\begin{array}{lll}
\langle \mathcal{M}, s_i\rangle \models \mathbf{start} & \text{iff} & i = 0 \\
\langle \mathcal{M}, s_i\rangle \models p & \text{iff} & p \in L(s_i) \\
\langle \mathcal{M}, s_i\rangle \models \mathbf{A}B & \text{iff} & \dot{\forall}_{\, x_{s_i}} .\ \langle \mathcal{M}, x_{s_i}\rangle \models B \\
\langle \mathcal{M}, s_i\rangle \models \mathbf{E}B & \text{iff} & \dot{\exists}_{\, x_{s_i}} .\ \langle \mathcal{M}, x_{s_i}\rangle \models B \\
\langle \mathcal{M}, x_{s_i}\rangle \models p & \text{iff} & \langle \mathcal{M}, s_i\rangle \models p \\
\langle \mathcal{M}, x_{s_i}\rangle \models \Box B & \text{iff} & \dot{\forall}_{\, s_j \in x_{s_i}} .\ (i \leq j) \ \dot{\Rightarrow} \\
 & & \langle \mathcal{M}, s_j\rangle \models B \\
\langle \mathcal{M}, x_{s_i}\rangle \models \Diamond B & \text{iff} & \dot{\exists}_{\, s_j \in x_{s_i}} .\ ((i \leq j) \ \dot{\wedge} \\
 & & \langle \mathcal{M}, s_j\rangle \models B) \\
\langle \mathcal{M}, x_{s_i}\rangle \models \bigcirc B & \text{iff} & \langle \mathcal{M}, s_{i+1}\rangle \models B (s_{i+1} \in x_{s_i}) \\
\langle \mathcal{M}, x_{s_i}\rangle \models A\,\mathcal{U}\,B & \text{iff} & \dot{\exists}_{\, s_j \in x_{s_i}} .(i \leq j) \ \dot{\wedge} \\
 & & \langle \mathcal{M}, s_j\rangle \models B) \ \dot{\wedge} \\
 & & \dot{\forall}_{\, s_k \in x_{s_i}} .(i \leq k < j) \ \dot{\Rightarrow} \\
 & & \langle \mathcal{M}, s_k\rangle \models A \\
\langle \mathcal{M}, x_{s_i}\rangle \models A\,\mathcal{W}\,B & \text{iff} & \langle \mathcal{M}, x_{s_i}\rangle \models \Box B \ \dot{\vee} \\
 & & \langle \mathcal{M}, x_{s_i}\rangle \models A\,\mathcal{U}\,B
\end{array}
$$

Figure 1 — CTL Semantics

Definition 1 *[Satisfiability] A well-formed CTL formula, B, is satisfiable if, and only if,* $\dot{\exists}\ \mathcal{M}.\ \langle \mathcal{M}, s_0\rangle \models B$.

Definition 2 *[Validity] A well-formed CTL formula, B, is valid if, and only if, it is satisfied in every possible model, i.e.* $\dot{\forall}\ \mathcal{M}.\ \langle \mathcal{M}, s_0\rangle \models B$.

3 Normal Form for CTL

As the proof method we use is clausal, we require both the definition of a normal form for CTL formulae and an algorithm for generating the normal form from arbitrary CTL formulae. To retain soundness, this algorithm must preserve satisfiability of the formulae. In developing the normal form, called Separated Normal Form for CTL (SNF_C), we utilise many of the mechanisms already developed for SNF in linear-time temporal logics [11]. Rather than repeating that algorithm, we will describe the elements that are new, being required for the manipulation of branching-time formulae. As in the linear-time case, the basic idea behind the normal form is to identify the core operators and generate formulae relevant to either the first state in a model, or to all subsequent states in a model. The transformation procedure again uses fixpoint unwinding and subformula renaming in order to reduce an arbitrary formula to SNF_C. A formula in SNF_C is of the form

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

where each P_i is a present-time formula and each F_i is a present or future-time formula.

$$
\begin{array}{lcll}
\mathbf{start} & \Rightarrow & \bigvee_{j=1}^{k} \beta_j & \text{(an } \textit{initial} \text{ rule)} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{A}\bigcirc \bigvee_{j=1}^{k} \beta_j & \text{(a } \textit{global}\, \mathbf{A}\Box\text{-rule)} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{E}\bigcirc \bigvee_{j=1}^{k} \beta_j \langle \text{ind}\rangle & \text{(a } \textit{global}\, \mathbf{E}\Box\text{-rule)} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{A}\Diamond\gamma & \text{(a } \textit{global}\ \mathbf{A}\Diamond\text{-rule)} \\
\bigwedge_{i=1}^{l} \alpha_i & \Rightarrow & \mathbf{E}\Diamond\gamma\langle \text{ind}\rangle & \text{(a } \textit{global}\ \mathbf{E}\Diamond\text{-rule)}
\end{array}
$$

Figure 2 — SNF_C Definition

Each of the "$P_i \Rightarrow F_i$" (called *rules*) is further restricted as in Figure 2, where each α_i, β_j or γ is a literal, **true** or **false** and $\langle \text{ind}\rangle$ is some index, possibly empty. The intuition behind this formulation is that each of the global rules represents a constraint upon the future behaviour of the formula, given the current conjunction of literals, while the initial rules provide the initial conditions; the '$\mathbf{A}\bigcirc$' and '$\mathbf{E}\bigcirc$' rules constrain successor states while the '$\mathbf{A}\Diamond$' and '$\mathbf{E}\Diamond$' constrain sequences of states over a longer period. All the other operators are subsumed within this representation. For example, the '$\mathbf{A}\Box$' operator is represented by a (possibly infinite) sequence of '$\mathbf{A}\bigcirc$' operations (see "removal of $\mathbf{A}\Box$" below).

3.1 Indices

An existential path quantifier plays a crucial role in CTL. In some cases we might have a formula $\mathbf{E}B$ derived from a formula $\mathbf{E}A$ and we want to be sure that both A and B hold on the same path. We incorporate indices to express this situation. Thus, $\mathbf{E}A\langle i\rangle$ means that A holds on some

path labelled as $\langle i \rangle$. As we will see, indices are important in the process of transformation to the normal form and in resolution where, for example, there is a need to merge formulae labelled with identical indices. Indices appear in any rule with a conclusion containing a **E**. In such cases both this premiss and conclusion(s) with **E** are labelled with an identical index. We also require that all rules applied retain indices. Note that we omit indices where they are empty.

3.2 Rules toward SNF_C

We now consider how an arbitrary CTL formula can be transformed into SNF_C. (Again, we will not provide transformations that occur within the analogous translation for linear-time logics [11].) The first transformation towards SNF_C is the introduction of the basic rule structure. Thus, given an arbitrary CTL formula, C, we can preserve satisfiability to give $\mathbf{A}\Box(\mathbf{start} \Rightarrow C)$.

Further transformations presented below are based upon fixpoint characterizations of the basic CTL modalities [5]:

$$\begin{array}{rcl} \mathbf{A}\Box F & \equiv & \nu\xi.\, \dot{F} \wedge \mathbf{A}\bigcirc\xi \\ \mathbf{A}(F\,\mathcal{U}\,G) & \equiv & \mu\xi.\, G \vee (F \wedge \mathbf{A}\bigcirc\xi) \\ \mathbf{E}\Box F & \equiv & \nu\xi.\, F \wedge \mathbf{E}\bigcirc\xi \\ \mathbf{E}(F\,\mathcal{U}\,G) & \equiv & \mu\xi.\, G \vee (F \wedge \mathbf{E}\bigcirc\xi) \end{array}$$

Given such equivalences we can develop our transformation rules. These are based on the general rules for representation of fixpoints for the basic modalities, for example

$$P \Rightarrow \nu\xi.\, Q(\xi) \quad \rightarrow \quad \left\{ \begin{array}{l} P \Rightarrow Q(x) \\ x \Rightarrow Q(x) \end{array} \right\}$$

where x is a new proposition symbol [11].

The specific rules for the removal of non-core modalities are given below. Note that formulae produced may still not be in the correct form and some further (generally classical) manipulation may be required.

Removal of $\mathbf{A}\Box$:

$$P \Rightarrow \mathbf{A}\Box F \rightarrow \left\{ \begin{array}{lcl} P & \Rightarrow & F \wedge x \\ x & \Rightarrow & \mathbf{A}\bigcirc(F \wedge x) \end{array} \right\}$$

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

$$P \Rightarrow \mathbf{E}\Box F\langle i \rangle \rightarrow \left\{ \begin{array}{lcl} P & \Rightarrow & F \wedge x \\ x & \Rightarrow & \mathbf{E}\bigcirc(F \wedge x)\langle i \rangle \end{array} \right\}$$

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

$$P \Rightarrow \mathbf{A}(F\,\mathcal{U}\,G) \rightarrow \left\{ \begin{array}{l} P \Rightarrow G \vee (F \wedge x) \\ x \Rightarrow \mathbf{A}\bigcirc(G \vee (F \wedge x)) \\ P \Rightarrow \mathbf{A}\Diamond G \end{array} \right\}$$

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

$$P \Rightarrow \mathbf{E}(F\,\mathcal{U}\,G)\langle i \rangle \rightarrow \left\{ \begin{array}{l} P \Rightarrow G \vee (F \wedge x) \\ x \Rightarrow \mathbf{E}\bigcirc(G \vee (F \wedge x))\langle i \rangle \\ P \Rightarrow \mathbf{E}\Diamond G\langle i \rangle \end{array} \right\}$$

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

$$P \Rightarrow \mathbf{A}(F\,\mathcal{W}\,G) \rightarrow \left\{ \begin{array}{l} P \Rightarrow G \vee (F \wedge x) \\ x \Rightarrow \mathbf{A}\bigcirc(G \vee (F \wedge x)) \end{array} \right\}$$

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

$$P \Rightarrow \mathbf{E}(F\,\mathcal{W}\,G)\langle i \rangle \rightarrow \left\{ \begin{array}{l} P \Rightarrow G \vee (F \wedge x) \\ x \Rightarrow \mathbf{E}\bigcirc(G \vee (F \wedge x))\langle i \rangle \end{array} \right\}$$

Recall that any application of a rule which contains **E** in its conclusion requires labelling **E** formulae in this conclusion and in its premiss with an identical index 'i'. Thus, in particular, in the case of the removal of $\mathbf{E}\mathcal{U}$ now we are sure that $\Diamond G$ will occur on the same path where $F\,\mathcal{U}\,G$ occurred.

We also require renaming for any formula with nested basic CTL modalities. An example of renaming is given below with x as a new variable:

$$\{P \Rightarrow \mathbf{E}(Q\,\mathcal{U}\,(\mathbf{E}F\,\mathcal{U}\,G))\} \rightarrow \left\{ \begin{array}{lcl} P & \Rightarrow & \mathbf{E}(Q\,\mathcal{U}\,x) \\ x & \Rightarrow & \mathbf{E}(F\,\mathcal{U}\,G) \end{array} \right\}$$

Now we present a rule to rewrite into SNF_C formulae of the type $Q \Rightarrow P$ with Q and P as pure classical formulae:

$$\{Q \Rightarrow P\} \rightarrow \left\{ \begin{array}{lcl} \mathbf{start} & \Rightarrow & \neg P \vee Q \\ \mathbf{true} & \Rightarrow & \mathbf{A}\bigcirc(\neg P \vee \neg Q) \end{array} \right\}$$

We also have to enrich the set of rules used for the transformation to the normal form in classical logic by the following (we term this set of transformation rules '*TRAN*'):

$$\begin{array}{rcl} \neg\mathbf{A}F & \rightarrow & \mathbf{E}\neg F \\ \neg\mathbf{E}F & \rightarrow & \mathbf{A}\neg F \\ \mathsf{P}\neg(F\,\mathcal{U}\,G) & \rightarrow & \mathsf{P}\neg G\,\mathcal{W}\,(\neg F \wedge \neg G) \\ \mathsf{P}\neg(F\,\mathcal{W}\,G) & \rightarrow & \mathsf{P}\neg G\,\mathcal{U}\,(\neg F \wedge \neg G) \\ Q \Rightarrow \mathsf{PT}(\mathbf{false}) & \rightarrow & Q \Rightarrow \mathbf{false} \\ Q \Rightarrow \mathbf{E}\bigcirc(P \wedge R)\langle i \rangle & \rightarrow & \left\{ \begin{array}{lcl} Q & \Rightarrow & \mathbf{E}\bigcirc P\langle i \rangle \\ Q & \Rightarrow & \mathbf{E}\bigcirc R\langle i \rangle \end{array} \right\} \end{array}$$

'P' abbreviates either path and 'T' any temporal operator.

As in the case of linear-time SNF, we can show that this transformation process *preserves satisfiability*.

4 The Temporal Resolution Method

Once the translation to SNF_C has been carried out, then all temporal statements within CTL are represented as sets of such rules. In order to achieve a refutation, we can apply two types of resolution rule: *step* resolution (SRES) and *temporal* resolution (TRES).

4.1 Step Resolution

Step (classical) resolution [10] can be used between formulae that refer to the *same* moment in time (here, l is a literal and $\langle i \rangle$ is an index):

SRES 1

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

SRES 2

$$\left\{ \begin{array}{l} P \Rightarrow \mathbf{E}\bigcirc(C \vee l)\langle i \rangle \\ Q \Rightarrow \mathbf{A}\bigcirc(D \vee \neg l) \end{array} \right\} \rightarrow P \wedge Q \Rightarrow \mathbf{E}\bigcirc(C \vee D)\langle i \rangle$$

SRES 3

$$\left\{ \begin{array}{l} P \Rightarrow \mathbf{E}\bigcirc(C \vee l)\langle i \rangle \\ Q \Rightarrow \mathbf{E}\bigcirc(D \vee \neg l)\langle i \rangle \end{array} \right\} \rightarrow P \wedge Q \Rightarrow \mathbf{E}\bigcirc(C \vee D)\langle i \rangle$$

SRES 4

$$\left\{ \begin{array}{l} \mathbf{start} \Rightarrow C \vee l \\ \mathbf{start} \Rightarrow D \vee \neg l \end{array} \right\} \rightarrow \{\mathbf{start} \Rightarrow C \vee D\}$$

Note that we again use indices in the second and third step resolution rules. This will be useful, in particular, in allowing merging formulae labelled with identical indices (see §4.4).

4.2 Temporal Resolution

Once step resolution has been applied, the temporal resolution rule can be invoked. The basic idea here is to resolve a formula containing a $\Box F$ constraint together with another containing a $\Diamond\neg F$ constraint. It is important to note that both of these must occur on the *same* path. Thus, we have four kinds of possible combinations of $\Box F$ and $\Diamond\neg F$ prefixed by **A** or **E**, taking into account even the case of $P \Rightarrow \mathbf{E}\Box F$ and $Q \Rightarrow \mathbf{E}\Diamond\neg F$, where the same path considered in both formulae. This generates four temporal resolution rules. Note that in the formulation of these rules we use the expressions $P \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\Box F$ and $P \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box F$ as abbreviations for $\mathrm{SNF_C}$ rules which represent these loops (see §4.3).

TRES 1

$$\left\{ \begin{array}{l} P \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\Box F \\ Q \Rightarrow \mathbf{A}\Diamond\neg F \end{array} \right\} \rightarrow \{\ Q \Rightarrow \mathbf{A}(\neg P\, \mathcal{W}\, \neg F)\ \}$$

TRES 2

$$\left\{ \begin{array}{l} P \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\Box F \\ Q \Rightarrow \mathbf{E}\Diamond\neg F\ \langle i \rangle \end{array} \right\} \rightarrow \{\ Q \Rightarrow \mathbf{E}(\neg P\, \mathcal{W}\, \neg F)\ \langle i \rangle\ \}$$

TRES 3

$$\left\{ \begin{array}{l} P \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box F \\ Q \Rightarrow \mathbf{A}\Diamond\neg F \end{array} \right\} \rightarrow \{\ Q \Rightarrow \mathbf{A}(\neg P\, \mathcal{W}\, \neg F)\ \}$$

TRES 4

$$\left\{ \begin{array}{l} P \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box F\ \langle i \rangle \\ Q \Rightarrow \mathbf{E}\Diamond\neg F\ \langle i \rangle \end{array} \right\} \rightarrow \{\ Q \Rightarrow \mathbf{E}(\neg P\, \mathcal{W}\, \neg F)\ \langle i \rangle\ \}$$

Here, the resolvent includes a $\mathcal{W}$ formula since there might be a case where there is a path x_{s_i} on which P occurs but $\Diamond\neg F$ is already outstanding.

4.3 Loops in CTL

While the idea of temporal resolution is to resolve formulae containing '$\Box$' and '$\Diamond$' operators, the actual procedure is more complex since in a number of difficult cases '$\Box$' formula can be presented in a given set of formulae containing '$\bigcirc$'. The following set of rules represents the idea of a simple example of such a *loop*:

$$\begin{array}{rcl} F & \Rightarrow & \mathbf{A}\bigcirc G \\ F & \Rightarrow & \mathbf{A}\bigcirc F \end{array}$$

This actually gives the so-called $\mathbf{A}\Box$-loop: $F \Rightarrow \mathbf{A}\bigcirc\mathbf{A}\Box G$, and thus could be resolved with, for example, $H \Rightarrow \mathbf{E}\Diamond\neg G$.

Thus, temporal resolution involves collecting together a set of formulae containing the '$\bigcirc$' operator that together imply the appropriate '$\Box$' formula. In particular cases, some of the formulae needed in order to construct a loop can be missed, but can be derived from the normal form representation. Thus, we require the removal of any '$\Box$' operator as a part of the transformation process into the normal form.

In the case of linear-time temporal logic, the crucial part of this resolution method is this process of searching for suitable sets of $\bigcirc$-formulae. In particular, a variety of algorithms have been developed and analysed for the implementation of this element of the method [10, 3, 4]. While we do not consider the implementation of the resolution method here, we expect that several of the successful algorithms from the linear-time case can be utilised within this framework (see also §6).

4.4 Transferral and Merging Rules

As in the linear-time temporal resolution method, a transformation is available for transferring constraints, derived from local contradictions, to other states. Thus, having $P \Rightarrow$ **false** we transform it as a pure classical formula using the rule mentioned in §3.2.

Searching for a loop might require merging formulae. This is presented by the following rule:

$$\left\{ \begin{array}{rcl} R & \Rightarrow & \mathsf{A}\bigcirc C \\ Q & \Rightarrow & \mathsf{P}\bigcirc D \end{array} \right\} \rightarrow \{R \wedge Q \Rightarrow \mathsf{P}\bigcirc(C \wedge D)\}$$

where 'P' represents either path quantifier. We even allow merging of two existential formulae with identical labels, as these identical labels indicate the same path:

$$\left\{ \begin{array}{l} P \Rightarrow \mathbf{E}\bigcirc C\langle i\rangle \\ Q \Rightarrow \mathbf{E}\bigcirc D\langle i\rangle \end{array} \right\} \rightarrow \{P \wedge Q \Rightarrow \mathbf{E}\bigcirc(C \wedge D)\langle i\rangle\}$$

4.5 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 any of the following $\mathrm{SNF_C}$ rules are generated during the temporal resolution procedure, the original set of rules is unsatisfiable:

1. $\mathbf{start} \Rightarrow \mathbf{false}$ 2. $\mathbf{true} \Rightarrow \mathbf{false}$

4.6 Correctness

Due to lack of space we will not provide full proofs here, but merely present a sketch of the proof of completeness and state the soundness and termination theorems.

Theorem 1 *[Soundness] If application of the resolution procedure to a well-formed formula, A, generates a contradiction, then A is unsatisfiable (in the given semantics for the logic CTL).*

Theorem 2 *[Termination] The resolution procedure terminates when applied to any well-formed formula.*

Theorem 3 *[Completeness] If a well-formed formula, A, is unsatisfiable, then the resolution procedure will generate a contradiction when applied to A.*

Completeness of our resolution system can be established by the following reasoning. We will show that for any theorem A of the axiomatic formulation of the logic CTL (see Appendix A) its negation $\neg A$ has a resolution refutation. As this resolution formulation of CTL is sound (Theorem 1 above) and the axiomatic formulation of CTL is complete (established in [5]) in relation to the same semantics, we are led to the completeness of the resolution system. Thus, for completeness we now have to show that our resolution formulation of CTL has at least the deductive power of the complete axiomatic system.

Lemma 1 *For any theorem, A, of the axiomatic formulation of the logic CTL its negation $\neg A$ has a resolution refutation.*

The proof below actually provides a constructive method for transforming any axiomatic proof into a refutation.

Proof We establish this fact by induction on the length n of the proof of A in the axiomatic system. Let $F_1, ..., F_n$ be a sequence of formulae within such a proof.

BASE CASE: $n = 1$, meaning that $F_1 = A$ is an axiom. Thus, we must show

B. Each axiom has a resolution-based refutation.

As an example, we present one part of the resolution proof for Ax5 (from the axiomatic system presented in Appendix A) by assuming its negation and deriving a contradiction. (This will, at the same time, serve as an example of the temporal resolution procedure in action.) The part of axiom Ax5 considered is

$$\mathbf{E}\Box p \Rightarrow \neg\mathbf{A}\Diamond\neg p$$

The refutation is given below, including the transformation (steps 1 to 9) of the negated formulae into the normal form.

1.	$\mathbf{start} \Rightarrow \mathbf{E}\Box p \wedge \mathbf{A}\Diamond\neg p$	*input*
2.	$\mathbf{start} \Rightarrow \mathbf{E}\Box p \ \langle i\rangle$	1, *TRAN*
3.	$\mathbf{start} \Rightarrow \mathbf{A}\Diamond\neg p$	1, *TRAN*
4.	$\mathbf{start} \Rightarrow p \wedge x$	2, $\mathbf{E}\Box$ Removal
5.	$x \Rightarrow \mathbf{E}\bigcirc(p \wedge x) \ \langle i\rangle$	2, $\mathbf{E}\Box$ Removal
6.	$\mathbf{start} \Rightarrow p$	4, *TRAN*
7.	$\mathbf{start} \Rightarrow x$	4, *TRAN*
8.	$x \Rightarrow \mathbf{E}\bigcirc p \ \langle i\rangle$	5, *TRAN*
9.	$x \Rightarrow \mathbf{E}\bigcirc x \ \langle i\rangle$	5, *TRAN*
10.	$\mathbf{start} \Rightarrow \mathbf{A}\neg x \mathcal{W} \neg p$	$3, 8-9$, *TRES3*
11.	$\mathbf{start} \Rightarrow \neg p \vee (\neg x \wedge y)$	10, $\mathbf{A}\mathcal{W}$ Removal
12.	$y \Rightarrow \mathbf{A}\bigcirc(\neg p \vee (\neg x \wedge y))$	10, $\mathbf{A}\mathcal{W}$ Removal
13.	$\mathbf{start} \Rightarrow (\neg x \wedge y)$	6, 11 *SRES 4*
14.	$\mathbf{start} \Rightarrow \neg x$	13, *TRAN*
15.	$\mathbf{start} \Rightarrow \mathbf{false}$	7, 14 *SRES 4*

Here "*TRAN*"and "*SRES*" are abbreviations for transformation and step resolution rules, respectively. Steps 4, 5, 11 and 12 represent the removal of $\mathbf{E}\Box$ and $\mathbf{A}\mathcal{W}$ operators. Note that the rules $8-9$ represent the loop $x \Rightarrow \mathbf{E}\bigcirc\mathbf{E}\Box p$ and hence can be resolved with the eventuality of the rule 3; this is represented by the *TRES3* operation at step 10. The proof terminates since it reached $\mathbf{start} \Rightarrow \mathbf{false}$. It is not very difficult to develop resolution proofs for the rest of the axioms, thus establishing statement *B*.

INDUCTION CASE: *Lemma 1* is valid for the proof of the length i, $1 \leq i < n$, i.e. for any i, F_i has a resolution proof. Now we must show that

I. *Lemma 1* is also valid for the proof of the length n. Considering the formula F_n, the following two cases can be indicated:

I.1. F_n is an axiom.

I.2. F_n is a conclusion of *generalization* or *modus ponens* inference rules.

In the case *I.1* the proof of the Lemma is identical to the base case of induction, *B*. Now consider the two subcases

of *I.2*.

I.2.1 Generalization (see Appendix A)

$F_n = \mathbf{A}\Box B$ where $\mathbf{A}\Box B$ is a conclusion of generalisation. Given as a premice the fact that B is valid (induction hypothesis), we will resolve this with the anchored negation of $\mathbf{A}\Box B$:

0.	$\mathbf{start} \Rightarrow \neg\mathbf{A}\Box B$	*given*
1.	$\mathbf{start} \Rightarrow \mathbf{E}\Diamond\neg B\langle i\rangle$	0, *TRAN*
2.	$\mathbf{start} \Rightarrow B$	*induction hypothesis*
3.	$\mathbf{true} \Rightarrow \mathbf{A}\bigcirc B$	*induction hypothesis*
4.	$\mathbf{start} \Rightarrow \mathbf{E}(\mathbf{false}\ \mathcal{W}\ \neg B)\langle i\rangle$	1, 3 *TRES2*
5.	$\mathbf{start} \Rightarrow \neg B$	4, Removal of $\mathbf{E}\mathcal{W}$
6.	$\mathbf{start} \Rightarrow \mathbf{false}$	2, 5 *SRES 4*

I.2.2 Modus ponens (see Appendix A)

Here we will use the fact (which can be easily proved for our resolution systems) that for any formula F, F has a refutation if $\mathbf{A}\Box F$ has. Thus, we have to show that $\neg\mathbf{A}\Box B$, together with induction hypothesis for the premises of the *modus ponens*, lead to a contradiction. This will prove (according to the statement above) the same result for the conclusion B of modus ponens itself. Now anchoring $\neg\mathbf{A}\Box B$ to **start**, we commence with

0.	$\mathbf{start} \Rightarrow \neg\mathbf{A}\Box B$	*given*
1.	$\mathbf{start} \Rightarrow \mathbf{E}\Diamond\neg B\langle i\rangle$	0, *TRAN*
2.	$\mathbf{start} \Rightarrow A$	*induction hypothesis*
3.	$\mathbf{start} \Rightarrow \neg A \vee B$	*induction hypothesis*
4.	$\mathbf{true} \Rightarrow \mathbf{A}\bigcirc A$	*induction hypothesis*
5.	$\mathbf{true} \Rightarrow \mathbf{A}\bigcirc(\neg A \vee B)$	*induction hypothesis*
6.	$\mathbf{true} \Rightarrow \mathbf{A}\bigcirc B$	4, 5 *SRES 1*
7.	$\mathbf{start} \Rightarrow \mathbf{E}(\mathbf{false}\ \mathcal{W}\ \neg B)\langle i\rangle$	1, 6 *TRES2*
8.	$\mathbf{start} \Rightarrow \neg B$	7, Removal of $\mathbf{E}\mathcal{W}$
9.	$\mathbf{start} \Rightarrow B$	2, 3 *SRES 4*
10.	$\mathbf{start} \Rightarrow \mathbf{false}$	8, 9 *SRES 4*

This concludes the outline proof of completeness.

5 Extensions of CTL

There are known extensions of CTL which can be generated from CTL by weakening its syntactic restrictions. Allowing Boolean combinations of temporal operators, but retaining CTL restrictions on path operators, gives us the CTL$^+$ system. Thus, a formula $\mathbf{E}(\Diamond A \vee \Box\neg A)$ is a formula of CTL$^+$ but not of CTL.

Another way to extend CTL is to introduce the $\Diamond^\infty$, meaning "infinitely often" and $\Box^\infty$, meaning "almost always", operators while retaining all the syntactic restrictions of CTL. This provides the Extended CTL (ECTL) system [8], which allows the representation of fairness properties. ECTL can be also enriched to by allowing boolean combinations of linear modalities, which gives a system called ECTL$^+$.

Recall that it has been shown that CTL$^+$ is equivalent to its base logic CTL, ECTL is strictly more expressive than both CTL [7] and ECTL$^+$ and that the most powerful branching-time system of this class is CTL* (which allows all possible combinations of modalities) [9].

5.1 Extended CTL

In this section, we consider ECTL and the application of our resolution procedure to this extended logic. Throughout, we will only describe the differences between the CTL procedure and the ECTL one. We begin by examining the syntactic and semantic extensions of ECTL.

We introduce two new temporal operators, namely

$\Box^\infty$ – 'almost always in the future'
$\Diamond^\infty$ – 'infinitely often in the future'

and extend the notion of well-formed formulae given in §2 to include

5. If A is in WFF_{ECTL}, then $\Box^\infty A$ and $\Diamond^\infty A$ are both *path* formulae.

The semantics of these two operators is as follows.

$$\langle \mathcal{M}, x_{s_i}\rangle \models \Diamond^\infty A \quad \text{iff} \quad \dot{\forall}_{s_j \in x_{s_i}}.(i \leq j) \Rightarrow \langle \mathcal{M}, s_j\rangle \models \Diamond A$$

$$\langle \mathcal{M}, x_{s_i}\rangle \models \Box^\infty A \quad \text{iff} \quad \dot{\exists}\ s_j \in x_{s_i}.(i \leq j) \wedge \langle \mathcal{M}, s_j\rangle \models \Box A$$

Now, to incorporate ECTL formulae in our resolution procedure it turns out that we need only translate the relevant operator pairs into SNF_C, using the following transformations.

$$P \Rightarrow \mathbf{E}\Diamond^\infty F\langle i\rangle \rightarrow \{P \Rightarrow x,\ x \Rightarrow \mathbf{E}\Diamond(x \wedge F)\}\langle i\rangle$$

$$P \Rightarrow \mathbf{A}\Diamond^\infty F \rightarrow \{P \Rightarrow x,\ x \Rightarrow \mathbf{A}\Diamond(x \wedge F)\}$$

$$P \Rightarrow \mathbf{E}\Box^\infty F\langle i\rangle \rightarrow \{P \Rightarrow \mathbf{E}\Diamond y,\ y \Rightarrow \mathbf{E}\Box F\}\langle i\rangle$$

$$P \Rightarrow \mathbf{A}\Box^\infty F \rightarrow \{P \Rightarrow \mathbf{A}\Diamond y,\ y \Rightarrow \mathbf{A}\Box F\}$$

Thus, SNF_C is expressive enough to serve as a normal form for both CTL and ECTL. The proof of completeness for ECTL is consequently very similar, the only difference being that we must now show that the transformation of WFF_{ECTL} to the normal form preserves satisfiability.

6 Conclusions

We have described an extension of 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 further extension towards the logic CTL*, which allows arbitrary combinations of path and temporal operators. During

this future work, we expect not only to refine both the normal form and the resolution rules, but also to provide further insight into the relationship between branching-time temporal logics by studying the formulae that they correspond to within the normal form.

Acknowledgements

Both authors would like to thank Clare Dixon for her comments and suggestions on this work.

References

[1] E. M. Clarke and E. A. Emerson. Design and Synthesis of Synchronisation Skeletons Using Branching Time Temporal Logic. *Proceedings of the Workshop on Logic of Programs*, Lecture Notes in Computer Science, 131:52–71, 1981.

[2] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[3] C. Dixon. *Strategies for Temporal Resolution*. PhD thesis, Department of Computer Science, University of Manchester, Manchester M13 9PL, U.K., 1995.

[4] C. Dixon. Search Strategies for Resolution in Temporal Logics. In *Proceedings of the Thirteenth International Conference on Automated Deduction*, 1996.

[5] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier, 1990.

[6] E. A. Emerson and J. Y. Halpern. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. In *Proceedings of the Fourteenth ACM Symposium on the Theory of Computing (STOC)*, pages 169–180, 1982.

[7] E. A. Emerson and J. Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time Temporal Logic. *ACM Journal*, 33(1):151–178, January 1986.

[8] E. A. Emerson and C-L. Lei. Modalities for Model Checking: Branching Time Logic strikes back. Technical Report TR–85–21, Department of Computer Sciences, University of Texas at Austin, 1985.

[9] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61, 1984.

[10] M. Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia, 1991.

[11] M. Fisher. A Normal Form for First-Order Temporal Formulae. In *Proceedings of Eleventh International Conference on Automated Deduction (CADE)*, 1992.

[12] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the Eighteenth Symposium on the Foundations of Computer Science*, Providence, USA, November 1977.

[13] A. Prior. *Past, Present and Future*. Oxford University Press, 1967.

[14] A. S. Rao and M. P. Georgeff. Modeling Agents within a BDI-Architecture. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, April 1991.

A Axiomatic System

The following set of axioms and rules of deduction represent a complete deductive system in relation to the semantics of the logic CTL. Here we follow [5].

Axioms for CTL

Ax1. All axioms of propositional calculus

Ax2. $\mathbf{E}\Diamond p \equiv \mathbf{E}(\mathbf{true}\ \mathcal{U}\ q)$

Ax3. $\mathbf{A}\Box p \equiv \neg\mathbf{E}\Diamond\neg p$

Ax4. $\mathbf{A}\Diamond p \equiv \mathbf{A}(\mathbf{true}\ \mathcal{U}\ p)$

Ax5. $\mathbf{E}\Box p \equiv \neg\mathbf{A}\Diamond\neg p$

Ax6. $\mathbf{E}\bigcirc(p \vee q) \equiv (\mathbf{E}\bigcirc p \vee \mathbf{E}\bigcirc q)$

Ax7. $\mathbf{A}\bigcirc p \equiv \neg\mathbf{E}\bigcirc\neg p$

Ax8. $\mathbf{E}(p\,\mathcal{U}\,q) \equiv (q \vee (p \wedge \mathbf{E}\bigcirc\mathbf{E}(p\,\mathcal{U}\,q)))$

Ax9. $\mathbf{A}(p\,\mathcal{U}\,q) \equiv (q \vee (p \wedge \mathbf{A}\bigcirc\mathbf{A}(p\,\mathcal{U}\,q)))$

Ax10. $\mathbf{A}\bigcirc\mathbf{true} \wedge \mathbf{E}\bigcirc\mathbf{true}$

Ax11. $\mathbf{A}\Box(r \supset (\neg q \wedge \mathbf{E}\bigcirc r)) \supset (r \supset \neg\mathbf{A}(p\,\mathcal{U}\,q))$

Ax12. $\mathbf{A}\Box(r \supset (\neg q \wedge \mathbf{E}\bigcirc r)) \supset (r \supset \neg\mathbf{A}\Diamond q)$

Ax13. $\mathbf{A}\Box(r \supset (\neg q \wedge \mathbf{A}\bigcirc r)) \supset (r \supset \neg\mathbf{E}(p\,\mathcal{U}\,q))$

Ax14. $\mathbf{A}\Box(r \supset (\neg q \wedge \mathbf{A}\bigcirc r)) \supset (r \supset \neg\mathbf{E}\Diamond q)$

Ax15. $\mathbf{A}\Box(p \supset q) \supset (\mathbf{E}\bigcirc p \supset \mathbf{E}\bigcirc q)$

Rules of Inference for CTL

For any theorems A and $A \Rightarrow B$,

R1. $A, A \supset B \vdash B$ *modus ponens*

R2. $B \vdash \mathbf{A}\Box B$ *generalization*