A Tableau for the Combination of CTL and BCTL*

John C. M^{c}Cabe-Dansted
School of Computer Science and Software Engineering
University of Western Australia
Perth, Australia
Email: john@csse.uwa.edu.au

***Abstract*—It is known that there is an exponential decision procedure for CTL. Given that important properties cannot be expressed in CTL, we seek a pure tableau based decision procedure (that does not rely on translations into automata) that is exponential for formulas that have only a bounded number of non-CTL properties. In this paper we present such a tableau for a combination of CTL and a bundled variant (BCTL*) of CTL* that is singly exponential for formulae with a bounded number of path-subformulae. The existing pure tableau for CTL* was built upon the pure tableau for BCTL*, so this paper is also a natural first step towards a pure tableau technique for CTL* that is singly exponential when similarly restricted.**

***Keywords*-Bundled; Exptime; Logic; Tableaux; Temporal;**

I. Introduction

There has been recent renewed interest in decision procedures for the branching time Full Computation Tree Logic (CTL*). It has long been known that CTL* is decidable and is 2EXPTIME complete, [1] provides a doubly exponential automaton based satisfiability checker, and [2] gives a lower-bound. These automaton based satisfiability checkers are expected to have performance close to their worst case on average and have not been implemented [3]. Recently, tableau based decision procedures have been proposed that have greater potential for reasonable real world performance, and that have publicly accessible implementations [3], [4]. However, the worst case performance is clearly still 2EXPTIME, whereas Computation Tree Logic (CTL) has a singly exponential decision procedure (see for example [5]). CTL is similar to CTL*, but the syntax of CTL is more restricted, as CTL pairs each temporal operator with a path-quantifier in such a way that the truth of a CTL formula does not depend on which of many possible futures occurs.

CTL is popular and can express many useful properties, unfortunately CTL cannot represent some important fairness related properties. Many interesting CTL* formulae are rather close to being CTL formulae, for example most of the sample formulae used in [4]. Thus is it natural to seek decision procedures that are exponential for formulas that are CTL-like, but that do not need to rigidly adhere to the syntactic restrictions of CTL.

The BCTL* logic (also known as $\forall LTFC$) is similar to the CTL* but instead of quantifying over all paths, instead quantifies over a bundle of paths. Although this bundle is suffix and fusion closed it need not be limit closed. For example, it may be the case that all paths include a right branch even though at every world there is a path where the next branch goes left; which violates the limit closure property. An argument for the 2-EXPTIME hardness of the decision problem could be made for BCTL* in a way similar to the argument for CTL* so from a computational complexity point of view, BCTL* is no easier to deal with than CTL*; however, the BCTL* logic is traditionally presented of being of theoretical interest as it is in some ways easier to reason with than CTL*; for example the specification for the tableau for BCTL* proposed by [6] was much simpler than the CTL* tableau [4] that was developed from it. Another example of BCTL* being easier (though not less computationally complex) to reason with was the discovery of a simple natural deduction system for a fragment of BCTL* [7].

In some cases we are interested only in futures that satisfy some fairness property. For example, when reasoning about some randomised algorithm it may be desirable to state that a fair coin could always come up either heads or tails but that any plausible future would not have an infinite series of tails. With BCTL*, we can construct a model with some fairness constraint on the bundle, while in CTL* the obvious attempt to formalise the fair coin would be a paradox. Every theorem of BCTL* is a theorem of CTL*. Proving a statement in BCTL* demonstrates that it is true not only in CTL* but would also be true if there was some form of fairness constraint on allowable paths.

Model checking formula CTL* with a bounded number of non-CTL properties is trivially polynomial. model checking. Model checking CTL* formulae is most naturally performed by recursively running a Linear Temporal Logic (LTL) model checker [8]. While model checking LTL is PSPACE in general, when the length of the input formula is bounded the complexity it is linear. For this reason, it is clear that so long as we place any finite bound on the length of the path-formulae, that need to be sent to LTL model-checker, we can model check such CTL* formulae in time linear in the length of the formula (see for example [9], but note that their main result is subtly different). This is convenient, as it means that we do not need to rigidly adhere to any particular syntactic restriction of CTL* to get model checking performance asymptotically similar to that of CTL. For example, we can add any property that can be represented by a CTL* formula ϕ to the syntax of CTL and the resulting language can still be model checked in polynomial time.

For formulae with a bounded number of non-CTL properties the decision procedure of [3] already runs in a singly exponential amount of time, as does the tableau for plain CTL of [5]. Both the pure tableau of [4], [6] and the hybrid tableau of [3] have to deal with sets of sets of formulae leading to doubly exponential running time in the worst case. In the case of [3] the sets of formulae are called "blocks" and represent a disjunction (or conjunction) of formulae that must hold on all (some) futures leaving from a particular state. It appears that it would be easy for the authors of [3] to show that CTL-like blocks have a unique derivation, and so if we limit the number of non-CTL subformulae we eliminate one exponential from the running time of the hybrid tableau. However, [3] do not explicitly make this claim or present such a proof.

Unlike our paper, [3] uses a hybrid automata based approach. The pure tableau based decision procedure of [6] for BCTL* requires a doubly exponential amount of time even when the non-CTL properties are bounded. These two tableau based techniques are very different and are expected to have different real-world running times. In particular, the requirement of the hybrid automata to build parity games may limit its ability to prove that large but simple formulae are satisfiable quickly [10]. As pure tableau work directly on subformulae of the formulae input by the user, the proofs generated by a pure tableau technique may be more meaningful to the user than a proof generated by translation into automata.

In this paper we present a tableau for deciding a combination of CTL and BCTL*. The combination of CTL and BCTL* allows this tableau to reason about combinations of bundled and unbundled properties (for examples of these, see Section II-D). We choose this combination for two reasons. Firstly, we can combine the tableau for CTL and BCTL* in a relatively natural way, unlike some other combinations we have considered. Secondly, it preserves EXPTIME-ness when the number of path-subformulae is bounded. The BCTL* tableau considered here can also be considered a simplification of the rather complex tableau for CTL* found in the 42 page paper [4].

We note that when only finite periods of time are considered the bundled and unbundled semantics are equivalent (see for example [11]). In this case, we can replace pairs of BCTL* operators with the corresponding CTL operators to minimise the number of path subformulae and maximise the performance of this tableau based decision procedure.

II. BCTL* and CTL

A. Syntax

Bundles affect the semantics rather than the syntax. CTL* and CTL have the same syntax as the corresponding BCTL* and BCTL logics.

Where p varies over $\mathcal{V}$ the set of variables (or atomic propositions), we define CTL* formulæ according to the following abstract syntax:

$$\phi := p \,|\, \neg\phi \,|\, (\phi \wedge \phi) \,|\, (\phi U \phi) \,|\, N\phi \,|\, A\phi \ .$$

The $\neg$, $\wedge$, N, U and A operators are read as "not", "and", "next", "until" and "all paths". We define the other operators as abbreviations: $\bot \equiv (p \wedge \neg p)$, $\top \equiv \neg\bot$, $\alpha \vee \beta \equiv \neg(\neg\alpha \wedge \neg\beta)$, "Finally" $F\alpha = \top U \alpha$, "Globally/Always" $G\alpha \equiv \neg F \neg\alpha$, "Weak Until" $\alpha W \beta \equiv \alpha U \beta \vee G\alpha$, "Exists a Path" $E\alpha \equiv \neg A \neg\alpha$, $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$ and $\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

The syntax of CTL is as follows:

$$\phi ::= p \,|\, \neg\phi \,|\, (\phi \wedge \phi) \,|\, A(\phi U \psi) \,|\, AN\phi \,|\, E(\phi U \psi) \ .$$

When combining CTL and BCTL* it can be ambiguous whether we are using the bundled or unbundled semantics. To address this, whenever using the unbundled semantics we will use underlining, so a CTL "AN" is written instead "$\underline{AN}$". To emphasise the difference further we will put the path quantifier together with the until operator, as in CTL they cannot be separated. Thus the CTL $A(\phi U \psi)$ will be written as $\phi \underline{AU} \psi$, and similarly for the E operator. The syntax of our combination of CTL and BCTL* is as follows:

$$\begin{aligned}\phi &:= \psi \,|\, \neg\phi \,|\, (\phi \wedge \phi) \,|\, (\phi U \phi) \,|\, N\phi \\ \psi &:= p \,|\, \neg\psi \,|\, (\psi \wedge \psi) \,|\, A\phi \,|\, \underline{AN}\psi \,|\, (\psi \underline{EU} \psi) \,|\, (\psi \underline{AU} \psi)\end{aligned}$$

Formulae of the form ψ are called state formulae. Formulae not of the form ψ are called path formulae. We use the following abbreviations for CTL formulae: $\underline{EN}\alpha \equiv \neg\underline{AN}\neg\alpha$, $\underline{EF}\alpha \equiv \top\underline{EU}\alpha$, $\underline{AF}\alpha \equiv \top\underline{AU}\alpha$, $\underline{EG}\alpha \equiv \neg\underline{AF}\neg\alpha$.

B. BCTL-Structures

In this section we will define a number of basic terms that we will then use to define BCTL-structures. Note that BCTL*-structures and BCTL-structures are the same, we will omit the "*" for aesthetic reasons and consistency.

Definition 1. We say that a binary relation R on S is **serial** (total) if for every a in S there exists b in S such that aRb. A **transition frame** is a pair $(W, \rightarrowtail)$, where W is a non-empty set of states and $\rightarrowtail$ is a serial relation on W.

Definition 2. A **valuation** g is a map from a set of states W to the power set $2^{\mathcal{V}}$ of the variables. The statement $p \in g(w)$ means roughly "the variable p is true at state w".

We will now formalize some notation relating to paths, and which sets of paths can be called bundles.

Definition 3. For any relation R we let R^* (respectively R^ω) be the set of finite (resp. infinite) paths through R. We call an ω-sequence $\sigma = \langle w_0, w_1, \ldots \rangle$ of states a **fullpath** iff $\sigma \in \rightarrowtail^\omega$, that is for all non-negative integers i we have $w_i \rightarrowtail w_{i+1}$. For all i in $\mathbb{N}$ we define $\sigma_{\geq i}$ to be the fullpath $\langle w_i, w_{i+1}, \ldots \rangle$, we define σ_i to be w_i and we define $\sigma_{\leq i}$ to be the sequence $\langle w_0, w_1, \ldots, w_i \rangle$. We say that a set of fullpaths B is **fusion closed** iff for all non-negative integers i, j and $\sigma, \pi \in B$ we have $\langle \sigma_0, \sigma_1, \ldots, \sigma_i, \pi_j, \pi_{j+1}, \ldots \rangle \in B$ if $\sigma_{i+1} = \pi_j$. We say that a set of fullpaths B is **suffix closed** iff for all integers i and $\sigma \in B$ we have $\sigma_{\geq i} \in B$. We say a set of fullpaths is a **bundle** if it is non-empty, suffix closed and fusion closed. We say a bundle B is on a transition frame $(W, \rightarrowtail)$ if $B \subseteq \rightarrowtail^\omega$ and every edge in $\rightarrowtail$ appears in B.

We complete this section with our definition of BCTL-structures.

Definition 4. A **BCTL-structure** $M = (W, \rightarrowtail, g, B)$ is a 4-tuple containing a set of states W, a serial binary relation $\rightarrowtail$ on W, a valuation g on the set of states W, and B is a bundle on $(W, \rightarrowtail)$.

C. Semantics

The semantics of the classical operators is similar to the definition in classical logic, although we will use fullpaths in place of worlds, that is:

$$M, \sigma \vDash \neg\phi \text{ iff } M, \sigma \nvDash \phi$$
$$M, \sigma \vDash \phi \wedge \psi \text{ iff } M, \sigma \vDash \phi \wedge M, \sigma \vDash \psi \ ,$$

Both CTL* and BCTL* use the operators from propositional Linear Temporal Logic (LTL),

$$M, \sigma \vDash N\phi \text{ iff } M, \sigma_{\geq 1} \vDash \phi$$
$$M, \sigma \vDash \phi U \psi \text{ iff } \exists i \in \mathbb{N} \text{ s.t. } M, \sigma_{\geq i} \vDash \psi$$
$$\text{and } \forall j \in \mathbb{N} \text{ s.t. } j < i \implies M, \sigma_{\geq j} \vDash \phi \ .$$

We define the semantics of the BCTL* Bundled All Paths operators A and the CTL* All Paths operator $\underline{A}$ as follows:

$$M, \sigma \vDash A\phi \text{ iff } \forall \pi \in B \text{ s.t. } \pi_0 = \sigma_0,\ M, \pi \vDash \phi$$
$$M, \sigma \vDash \underline{A}\phi \text{ iff } \forall \pi \text{ s.t. } \pi_0 = \sigma_0,\ M, \pi \vDash \phi \ .$$

The semantics of the CTL operators are defined in terms of the CTL* $\underline{A}$ operator and the LTL operators as follows: $\underline{AN}\psi \equiv \underline{A}N\psi$, $(\phi \underline{EU} \psi) \equiv \neg\underline{A}\neg(\phi U \psi)$, $(\phi \underline{AU} \psi) \equiv \underline{A}(\phi U \psi)$.

D. Examples of Formulae

Now that we have defined the logic, we can revisit the fair coin example. We can encode the idea that on all paths we expect to always finally have a head toss as $AGFh$ (and the reverse, that we always expect to have have a tail as $AGF\neg h$). We can encode the idea that it is always possible that the next toss is a tail by the formula $AGEN\neg h$.

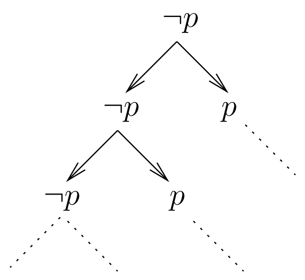

Figure 1. Structure on which $\underline{AF}p$ is false, but AFp can be true.

Although $AGFh \wedge AGEN\neg h$ is not satisfiable in CTL* it is satisfiable in BCTL*. This can be easy verified by, for example, entering the formula `(AGFh&AGEX-h)` into the BCTL* web applet [12].

Although a major reason this combination of BCTL* and CTL was chosen was as a stepping stone to finding an efficient pure tableau decision procedure for CTL*, we now give a number of example formulae that can be reasoned with using this tableau. One of the possible uses of this tableau is for testing intuitions of the difference between bundled and unbundled semantics. For example, we would expect $ENp \leftrightarrow \underline{EN}p$ to be valid as every edge in $\rightarrowtail$ appears in B. As B is suffix and fusion closed, we would expect $EFp \leftrightarrow \underline{EF}p$. Since every path in the bundle B is clearly a path, we would expect $EGp \rightarrow \underline{EG}p$ to also be valid. However, since the difference between bundled and unbundled logics is that the bundle B need not contain all paths, we would expect $\underline{EG}p \rightarrow EGp$ to be falsifiable. In addition to these trivial examples, the tableau we will define in this paper could also reason about more complex examples; such as verifying that $E(pUA(pUq)) \rightarrow (p\underline{EU}q)$ is valid. One example that is good for visualising the difference between bundled and unbundled semantics is $\neg\underline{AF}p \wedge AFp$. This would be satisfied on a binary tree where only the leftmost branch satisfies $G\neg p$. We can construct a bundle which only includes paths that follow the left edge a finite number of times, and so $\neg\underline{AF}p \wedge AFp$ is satisfiable (see Figure 1).

The tableau we will define can be used to efficiently reason about some BCTL* formulae. For a BCTL* formula ψ, let $\underline{\psi}$ represent the formula resulting when BCTL operators are replaced with CTL operators. In many cases ψ and $\underline{\psi}$ are equivalent (and we can test whether they are equivalent by testing whether $\psi \leftrightarrow \underline{\psi}$ is valid). Say we are attempting to determine whether a BCTL* formula ϕ is satisfiable. We can improve the performance of the decision procedure by recursively replacing each subformula ψ with $\underline{\psi}$ when these are known to be equivalent. In general, for any set Φ of BCTL* formula consisting only of BCTL* formulae which are computationally easy to transform into equivalent formulae with only a bounded number of path-subformulae,

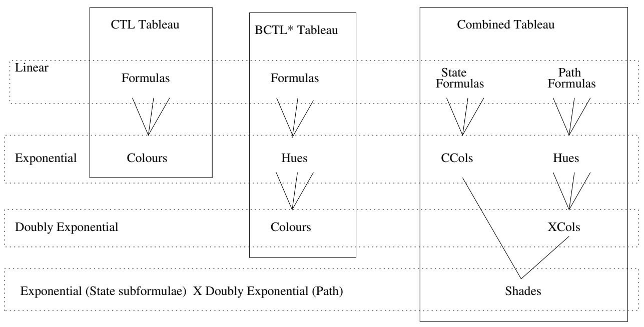

Figure 2. Comparison of Tableau

testing the satisfiability of the formulae in Φ is at worst singly exponential.

III. A Pre-Tableau for BCTL* and CTL

Here we define a tableau CTAB for deciding this combination of CTL and BCTL*. This tableau is derived from Reynolds' [6] tableau for BCTL* and the decision procedure for CTL defined in [5].

As this paper uses a number of terms, we show how these terms relate in Figure 2, and how they relate to the terms in the tableaux this paper extends. It is traditional to call the labels of nodes of a tableau colours. As is common colours of the existing CTL tableaux are sets of state formulae, we call this type of colour a CTL Colour (CCol). In BCTL* the truth of a formula can depend not only on the initial state σ_0 but also on the remainder of the path σ. For this reason the existing BCTL* tableau needs to consider not only what set of formulae are true along a particular path, but also what types of path start from each world. For this reason the colours of the BCTL* tableaux are sets of sets of formulae, we call this type of colour an eXtended colour (XCol). As we will be discussing a number of different types of colours, to reduce ambiguity we will refer the colours used to label the nodes of our new combined tableau as Shades. The shades are a combination of a CCol and an XCol.

We now fix a formula ϕ that we are attempting to determine the satisfiability of. Note that α is satisfiable iff $E\alpha$ is satisfiable, so we can assume without loss of generality that ϕ is a state formula.

An important tool in developing a tableau technique that requires a finite amount of space is identifying a finite set of formulae, called the closure set, such that the technique never has to reason about formulae outside this set.

Definition 5. We use $\alpha \leq \beta$ to indicate that α is a subformula of β, and we consider two formulae to be equal if they have the same representation (e.g. $p \wedge q = p \wedge q \neq q \wedge p$). The closure $\mathbf{cl}\phi$ of the formula ϕ is defined as the smallest set that satisfies the following requirements:

(Cl1) $\phi \in \mathbf{cl}\phi$
(Cl2) For all $\psi \in \mathbf{cl}\phi$, if $\delta \leq \psi$ then $\delta \in \mathbf{cl}\phi$.
(Cl3) For all $\psi \in \mathbf{cl}\phi$, if ψ is not of the form $\neg\delta$ then $\neg\psi \in \mathbf{cl}\phi$.
(Cl4) For all $(\alpha \underline{AU} \beta) \in \mathbf{cl}\phi$ we have $\underline{AN}(\alpha \underline{AU} \beta) \in \mathbf{cl}\phi$.
(Cl5) For all $(\alpha \underline{EU} \beta) \in \mathbf{cl}\phi$ we have $\underline{EN}(\alpha \underline{EU} \beta) \in \mathbf{cl}\phi$.

For simplicity, we will only consider sets of formulae that are Maximally Propositionally Consistent, as defined below.

Definition 6. We say that $h \subseteq \mathbf{cl}\phi$ is Maximally Propositionally Consistent **MPC** iff for all $\alpha, \beta \in \mathbf{cl}\phi$

(M1) if $\beta = \neg\alpha$ then $\beta \in h$ iff $\alpha \notin h$.
(M2) if $\alpha \wedge \beta \in \mathbf{cl}\phi$ then $(\alpha \wedge \beta) \in h \leftrightarrow (\alpha \in h$ and $\beta \in h)$.

Since the BCTL* tableau requires doubly exponential time, we wish to avoid using the BCTL* half of the combined tableau technique when possible. However, the BCTL* part is required to reason about path formula, and to do so it has to be aware of the truth of direct subformulae of path-formulae. We call the formulae that need to be considered by the BCTL* part of the combined tableau path-sensitive formulae.

Definition 7. We define the set P of **path-sensitive** formulae as the minimal set that satisfies the following:

(P1) If $\psi \in \mathbf{cl}\phi$ and ψ is a path formula then $\psi \in P$.
(P2) if $N\psi \in \mathbf{cl}\phi$ then $\psi \in P$ and $N\psi \in P$.
(P2) if $\psi U \theta \in \mathbf{cl}\phi$ then $\psi U \theta \in P$, $\psi \in P$ and $\theta \in P$.

A hue is roughly speaking a set of formulae that could plausibly hold along a single fullpath. As mentioned in [6] some hues are not satisfied on any fullpath, but every path corresponds to some hue. After defining hues we will define XCols (sets of hues roughly representing paths that could start at the same world), and CCols (representing state formulae that could plausibly hold at the same state).

Definition 8. A set $h \subseteq P$ of path-sensitive formulae is a **hue** for ϕ iff

(H1) h is MPC;
(H2) if $\alpha U \beta \in h$ then $\alpha \in h$ or $\beta \in h$;
(H3) if $\neg(\alpha U \beta) \in h$ then $\beta \notin h$; and
(H4) if $A\alpha \in h$ and $\alpha \in P$ then $\alpha \in h$.

Definition 9 (R^h_N). The temporal successor R^h_N relation on hues below is defined as in Reynolds [6]; For all hues a, b put (a, b) in R^h_N iff the following conditions are satisfied:

(R1) $N\alpha \in a$ implies $\alpha \in b$.
(R2) $\neg N\alpha \in a$ implies $\alpha \notin b$.
(R3) $\alpha U \beta \in a$ and $\beta \notin a$ implies $\alpha U \beta \in b$.
(R4) $\neg(\alpha U \beta) \in a$ and $\alpha \in a$ implies $\neg(\alpha U \beta) \in b$.

Definition 10. We call a set of hues a **XCol**. We define a temporal successor function R^X_N on XCols as follows: given a pair of XCols X and Y, we have XR^X_NY iff

(X1) for all hues $g \in X$ there exists $h \in Y$ such that gR^h_Nh.

Definition 11. A set of state formulae a is a **CCol** of ϕ iff

(C1) $a \subseteq \mathbf{cl}\phi$ and;
(C2) a is MPC;
(C3) if $\alpha\underline{AU}\beta \in a$ then $\alpha\underline{EU}\beta \in a$;
(C4) if $\alpha\underline{EU}\beta \in a$ then $\alpha \in a$ or $\beta \in a$;
(C5) if $\neg(\alpha\underline{AU}\beta) \in a$ then $\neg\beta \in a$.

Let $\mathcal{C}$ be the CCols of ϕ. We define a successor relation on $\mathcal{C}$ as follows:

Definition 12 (R^C_N)**.** We define a temporal successor relation R^C_N on CCols as follows: for all $C, D \in \mathcal{C}$, put $(C, D) \in R^C_N$ iff

(CN1) For all $\alpha\underline{AU}\beta \in C$, either $\beta \in C$ or $\alpha\underline{AU}\beta \in D$.
(CN2) For all $\neg(\alpha\underline{EU}\beta) \in C$, either $\neg\alpha \in C$ or $\neg(\alpha\underline{EU}\beta) \in D$.
(CN3) For all $\underline{AN}\alpha \in C$, we have $\alpha \in D$.

Note that is CN2 above, we do not need to explicitly state that $\neg\beta \in C$, as that is ensured by C5.

Definition 13. We call $Z = (C, X)$ a **shade** if X is a XCol and C is a CCol which satisfy the following conditions.

(Z1) For every hue h in X and for all state formulae α in P we have $\alpha \in h$ iff α in C.
(Z2) If $A\psi \in C$ then for all $h \in X$, we have $\psi \in h$.
(Z3) If $\neg A\psi \in C$ then there exists $h \in X$ such that $\neg\psi \in h$.
(Z4) If p is a path-sensitive variable, then for all $h \in X$, we have $p \in h$ iff $p \in C$.

Definition 14 (R^Z_N)**.** We define a temporal successor relation R^Z_N on shades as follows: for all pairs of CCols C, D and XCols X, Y we put $((C, X), (D, Y)) \in R^Z_N$ iff $(C, D) \in R^C_N$ and $(X, Y) \in R^X_N$.

Definition 15. Let W be a set of nodes, $\rightarrowtail$ be a binary relation on W and the function L from W to shades be a labelling of W with shades. Then we say $\langle W, \rightarrowtail, L\rangle$ is a **pre-tableau** iff for all u, v in W, we have $u \rightarrowtail v \implies L(u) R^Z_N L(v)$. We call the pre-tableau $\langle W, R^Z_N, I\rangle$ CTAB$_0$ when W is the set of shades and I is the identity function on W.

IV. Pruning the Tableau

We now show how to prune the tableau.

We now fix a pre-tableau $\langle W, \rightarrowtail, L\rangle$. For shorthand we define $(u_\rightarrowtail)$ to be the set of all successors of u, that is all $v \in W$ such that $u \rightarrowtail v$. We let $\{L\}(u_\rightarrowtail)$ be the set of all labels of $(u_\rightarrowtail)$, that is $Z \in \{L\}(u_\rightarrowtail)$ iff there exists $v \in u_\rightarrowtail$ such that $L(v) = Z$.

Informally, it is obvious that if say $\underline{EN}p$ occurs in a node but we cannot find a temporal successor p then there is something wrong, and that to fix this we need to remove the original node.

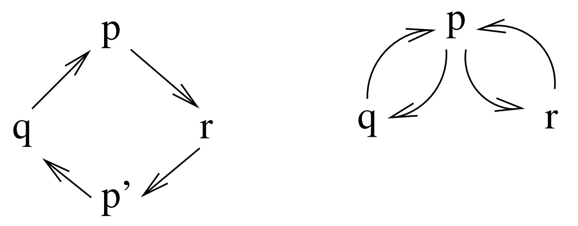

Figure 3. Full vs. Pseudo Hintikka Structure.

Definition 16. We say that a node w labelled with the shade (C, X) is **stepwise-unfulfilled** if we cannot find successors for every hue in X and every state formula in C. That is w is stepwise-unfulfilled if:

1) there exists a hue x in X such that for every $(D, Y) \in \{L\}(w_\rightarrowtail)$ and every $y \in Y$, the pair $(x, y) \notin R^h_N$; or
2) there exists a formula of the form $\neg\underline{AN}\alpha$ in C such that for every $(D, Y) \in \{L\}(w_\rightarrowtail)$ we have $\alpha \in D$.

Note the difference between R^Z_N and being stepwise-unfulfilled above. The first implements properties that every successor must have, while the second implements properties that some successor must have.

Definition 17. A **frontier node** is a node w that has no successors, that is $w_\rightarrowtail$ is empty. An **interior node** is a node that is not a frontier node. A **fragment** is a pre-tableau such that every node w of the pre-tableau is either stepwise-fulfilled or a frontier node. We say that a pre-tableau $\langle W, \rightarrowtail, L\rangle$ is a fragment of a pre-tableau $\langle W', \rightarrowtail', L'\rangle$ if $\langle W, \rightarrowtail, L\rangle$ is contained in $\langle W, \rightarrowtail^T, L^T\rangle$ a tree-unwinding of $\langle W', \rightarrowtail', L'\rangle$; that is if $\rightarrowtail \subseteq \rightarrowtail^T$ and $L(w) = L^T(w)$ for all $w \in W$.

In [5] they define Hintikka structures, which in essence provide a model for the formula under consideration. However, it is convenient to label every node in the tableau with a unique set (or set of sets) of formulae. This is easy with the BCTL* tableau of [6]; however, with CTL collapsing nodes with the same label can break a model. For example, say we have $(p \to AN\neg p) \wedge (\neg p \to ANp) \wedge AFq \wedge AFr$. Then we see that the structure on the left of Figure 3 models this formula and that the nodes p and p' satisfy the same subformulae and so they would have the same label. Thus collapsing duplicated labels as is done in [6] would result in the structure on the right which does not model the formula. The solution in [5] is to define pseudo-Hintikka structures, which do not model the formula, but which can be unwound into a model; the structure on the right can be unwound into a similar to the one on the left. The only difference between Hintikka structures and pseudo-Hintikka structures relates to the handling of formulae like AFq. In a pseudo-Hintikka structure we do not require that the structure actually satisfy AFq we only require that we can find some fragment of the structure that can satisfy AFq.

Definition 18. We say that a hue h is **inside** a node labelled with a shade $Z = (C, X)$ if $h \in X$. We say that a formula ϕ is inside the node if $\phi \in C \cup (\bigcup X)$.

Definition 19. We say that a node w labelled with a shade $Z = (C, X)$ is pseudo-Hintikka-fulfilled (**pHf**) if it is stepwise-fulfilled and satisfies the following three requirements:

(pHf1) for every hue h in X and formula of the form $\alpha U \beta \in h$ there exists a sequence of nodes $w, w_1, \ldots, w_m \in (\rightarrowtail^*)$ and sequence of hues $h, h_1, \ldots, h_m \in \left(R_N^h\right)^*$ such that:

1) the sequence fulfils $\alpha U \beta$, that is, $\beta \in h_m$.
2) each hue h_i is inside the node w_i.

(pHf2) For every formula of the form $(\alpha \underline{EU} \beta)$ in C there exists a sequence of nodes $w, w_1, \ldots, w_m \in (\rightarrowtail^*)$ such that β is inside w_m.

(pHf3) For every formula of the form $(\alpha \underline{AU} \beta)$ in C there exists a fragment of Z such that the CCols in the frontier shades of the fragment all contain β and the interior shades all contain α. (Note that we can use König's lemma to show that this fragment is finite.)

Definition 20. We say that a pre-tableau is a **tableau** iff all its nodes are pHf, when S' is taken to be the set of all nodes in the tableau.

The decision procedure is as follows: we begin with the pre-tableau CTAB$_0$ from Definition 15. We iteratively remove all nodes that are not pHf until all nodes are pHf. Note that removing one node may cause another node to become non-pHf, and so one pass is not sufficient. We say that CTAB succeeds if the resulting tableau contains a node labelled with a shade (C, X) where $\phi \in C$.

V. Complexity

Let m be the number of path-subformulae of ϕ. We see the number of path-sensitive formulae is at most $3m$. This is because each sensitive formula is either a path formula or a direct subformula of ϕ, and a formula has at most two direct subformulae ($\alpha U \beta$ has α and β as direct subformulae). Since hues are power-sets of sensitive formulae, the number of hues is a most 2^{3m}. Likewise, the XCols are power-sets of hues and we have at most $2^{2^{3m}}$ XCols.

Let n be the number of state-subformulae. The number of CCols is at most 2^{kn} for some constant k. A shade is a combination of a CCol and a XCol. Thus the number of shades is at most $2^{kn} 2^{2^{3m}}$. Note that each node in CTAB$_0$ is labelled with a unique shade, so it has at most $2^{kn} 2^{2^{3m}}$ nodes. This is singly exponential when the number of path subformulae is bounded (or of order $\ln n$).

For a sketch of how to test that a fragment satisfying pHf3 exists in time polynomial to the number of nodes/shades, see [5]. The remainder of the tests are clearly polynomial. As with other tableaux, we expect the average case performance to be much better than the theoretical worst case performance. This worst case bound suggests that the performance of the tableau will be similar to the tableau for CTL when the number of path subformulae is bounded and small.

VI. Soundness

CTAB is sound, that is, if it succeeds on ϕ then ϕ is satisfiable.

We will now show how to construct a model for the formula from the tableau CTAB. The details are as in [5] and [6]. Similarly to [5] (but unlike [6]) we will have to unwind the tableau into a tree to ensure that formulae such as $AF\alpha$ are satisfied at the worlds corresponding to nodes w where $AF\alpha \in w$. We assume some arbitrary ordering on the shades, fragments and formulae.

Consider a tableau $\langle S, R, L \rangle$ where S is the set of nodes, each labelled with a shade, the relation R forms a tree and all interior nodes of the tree are stepwise-fulfilled.

We define an eventuality in the tree as a tuple $\langle w, h, \psi \rangle$ where w is a node in the tree and ψ is a formula of the form $\alpha \underline{AU} \beta$, $\alpha \underline{EU} \beta$, $\alpha U \beta$. We say $\langle w, h, \alpha \underline{AU} \beta \rangle$ is fulfilled if β is inside some node of every path starting at w; note that as the interior nodes are stepwise-fulfilled we do not need to consider α in our definition of fulfilled, as we know that α will remain in the tableau until we reach β. We say $\langle w, h, \alpha \underline{EU} \beta \rangle$ is fulfilled if β is inside some node reachable from w. The definition of fulfilled for $\langle w, h, \alpha U \beta \rangle$ is similar but we have to follow the hues, or formally there must exist a sequence of nodes $w_0, w_1, \ldots, w_m$ through the tree and hues $h_0, h_1, \ldots, h_m$ that form a path through R_N^h such that for each non-negative integer $i \leq m$ we have h_i inside w_i, have $w_0 = w$ and $h_0 = h$.

We will now define an unwinding of CTAB into a tree. Let S' be the set of labels of the nodes in CTAB, that is the shades remaining after pruning. Consider the following algorithm:

1) We start with $S^\dagger = \{w_0\}$ and $R^\dagger = \emptyset$, where w_0 is a node labelled with a shade $Z = (C, X)$ satisfying $\phi \in C$.
2) We navigate the tree breadth-wise. When we come across a frontier node w labelled with a shade $Z = (C, X)$ we consider the oldest unfulfilled $\underline{AU}$ or $\underline{EU}$ eventuality.

a) If there is no such eventuality, or the eventuality is of the form $\alpha U \beta$ or $\alpha \underline{EU} \beta$, then for each temporal successor Z' of Z (that is, each $Z' \in S'$ such that $(Z, Z') \in R_N^Z$) we add a successor to w labelled Z'.
b) If the eventuality is of the form $\alpha \underline{AU} \beta$, we add the first fragment satisfying pHf3.

We see that at each step of the algorithm, all of the interior nodes are stepwise fulfilled. Since there are a finite number of unfulfilled eventualities on each branch of the tree, and the algorithm iteratively fulfils the oldest eventuality first, each eventuality will be fulfilled. As the algorithm never ends, we define the final tree (S_T, R_T) as containing all nodes and edges that are ever added by the algorithm.

Where (S_T, R_T) is our final tree we define a BCTL-structure $(W, \rightarrowtail, g, B)$ as follows: the transition frame (W, R) is simply (S_T, R_T), and the valuation $g(w)$ of a world is precisely those atoms p that are inside w. We now define the set of bundled paths B, in a similar fashion to how they were defined in [6].

Definition 21. We call an ω-sequence $\langle (w_0, h_0), (w_1, h_1), \ldots \rangle$ a **thread** through S_T iff for all $i \geq 0$: each $w_i \in S_T$, each hue h_i is inside w_i, $(w_i, w_{i+1}) \in R_T$, and each $(h_i, h_{i+1}) \in R_N^h$. We say that this is a fulfilling thread iff for all $i \geq 0$ and for all formulae of the form $(\alpha U \beta)$ in h_i, there exists $j \geq i$ such that $\beta \in h_j$.

We include a fullpath $\sigma = \langle w_0, w_1, \ldots \rangle$ in B iff there exists a fulfilling thread $\langle (w_0, h_0), (w_1, h_1), \ldots \rangle$, and we say that this thread **justifies** σ being in B. It is easy to show that B is a bundle. Since every $\alpha U \beta$ eventuality is fulfilled it is easy to see every hue has a fulfilling thread.

Lemma 22. *For all ψ in $\mathbf{cl}\phi$, for all threads $\mu = \langle (w_0, h_0), (w_1, h_1), \ldots \rangle$ justifying $\sigma = \langle w_0, w_1, \ldots \rangle$ we have*

$$(W, \rightarrowtail, g, B), \sigma \vDash \psi \text{ iff } \psi \text{ is inside } w_0.$$

The proof of this lemma is rather mechanical, given the previous results it is easy to prove this recursively, see [6] and [5] for details. One minor point not covered in [6] or [5] is that the formulae in the [6] style hues and [5] style CCols need to be consistent, for example if we had $(q\underline{EU}p)\,U\,(p\underline{AU}q)$ in a hue of a node w then we would clearly want to have either $(p\underline{AU}q)$ or $(q\underline{EU}p)$ in the CCol of the same node. This is ensured by P2, P3 and Z1.

Theorem 23. *The tableau is sound, that is if the tableau succeeds, ϕ is satisfiable.*

Obvious from lemma above.

VII. Completeness

Lemma 24. *CTAB is complete, that is, if ϕ is satisfiable then CTAB halts and succeeds on ϕ.*

The tableau is finite so CTAB will halt.

Say that ϕ is satisfiable. Then there exists a BCTL-structure $(W, \rightarrowtail, g, B)$ and path π^0 in B such that $\pi^0 \vDash \phi$. We will define a translation ρ from worlds to shades, and show that for each world w in W, the shade $\rho(w)$ will not be pruned from the tableau. Hence the set S' of unpruned nodes will be non-empty when CTAB halts, and so CTAB will succeed.

Definition 25. We define a function $\mathfrak{h}$ on paths such that

$$\mathfrak{h}(\pi) = \{\alpha : \alpha \in \mathbf{cl}\phi \text{ and } \pi \vDash \alpha\}$$

As H1–4 are simply properties that any set of formulae that hold along the same path must satisfy, it is clear that the following lemma holds.

Lemma 26. *From the semantics of BCTL*, we see that for each $\pi \in B$, $\mathfrak{h}(\pi)$ is a hue.*

Proof: (H1) Since the semantics of the $\wedge$ and $\neg$ operators in BCTL* come from classical logic, it is clear that $\mathfrak{h}(\pi)$ is MPC.

(H2) If $\alpha U \beta \in \mathfrak{h}(\pi)$ then $\pi \vDash \alpha U \beta$ and we see that either β is satisfied immediately and so $\pi \vDash \beta$ or $\pi \vDash \alpha$; hence $\alpha \in \mathfrak{h}(\pi)$ or $\beta \in \mathfrak{h}(\pi)$.

(H3) Likewise if $\neg(\alpha U \beta) \in \mathfrak{h}(\pi)$ then we see that $\pi \nvDash \alpha U \beta$ and so $\pi \nvDash \beta$ and so $\beta \notin \mathfrak{h}(\pi)$, demonstrating that H3 is satisfied.

(H4) If $A\alpha \in \mathfrak{h}(\pi)$ then $\pi \vDash A\alpha$ and so all paths starting at π_0, including π, satisfy α. ■

We will now define a function from worlds to shades. This definition uses the function from paths to hues defined in Definition 25.

Definition 27. We define a function ρ_X on worlds to sets of hues, ρ_C to sets of state formulae, ρ_Z to shades as follows:

$$\begin{aligned}
\rho_X(w) &= \{\mathfrak{h}(\pi)\colon\ \pi \in B \text{ and } \pi_0 = w\} \\
\rho_C(w) &= \{\alpha\colon\ \alpha \text{ is a state-formula in } \mathbf{cl}\phi \\
&\qquad ,\exists \sigma \in B \text{ with } \sigma_0 = w \wedge M, \sigma \vDash \alpha\} \\
\rho_Z(w) &= (\rho_C(w), \rho_X(w)) \\
\rho_S &= \{\rho_Z(w) : w \in W\}\ .
\end{aligned}$$

We see that for each $w \in W$, $\rho_X(w)$. Likewise $\rho_C(w)$ is a CCol and $\rho_Z(w) = (\rho_C(w), \rho_X(w))$ is a shade. We let the tableau be $\left(\rho_S, R_N^Z \cap (\rho_S \times \rho_S)\right)$. It is trivial to see the tableau is stepwise-fulfilled. Showing that the tableau is pHf is also easy, for details on pHf1 see [6] and see [5] for details on pHf2 and pHf3. Thus no nodes in ρ_S are pruned.

VIII. Conclusions and Future Research

We have presented a tableau for a combination of CTL and BCTL*. This tableau is singly exponential when the number of non-CTL operators is bounded; a pure CTL formula will not have any path subformulae. While this combination has some advantages, it is presented as a step towards finding a pure tableau for a similar combination of CTL and CTL*.

The BCTL* tableau used in this paper can be extended to a CTL* tableau, as was done in [4]. A simple replacement of the BCTL* with the tableau CTL* is not challenging though we note that the specification of the CTL* tableau alone is more lengthy than this paper. This expansion would provide better performance for CTL-like CTL* formulae; however a simple replacement may not preserve the singly exponential running time for CTL-like formulas. The worst case bound on the running time of the CTL* tableau is based on a bound on the size of models for CTL* like formulae. To preserve the singly exponential running time we would need to find a better halting condition.

Another important optimisation would be to convert these tableaux into conventional tableaux rooted with a single formula. This tableau begins by creating all possible shades. We define the tableau this way as it simplifies the definition and it does not affect that the worst-case performance results that are the focus of this paper. However, this form of tableaux for CTL and BCTL* also tends to perform much worse than conventional tableau in the average case [5], [10]. To see why, consider a formula of the form $p \wedge \neg p \wedge \phi$. A conventional tableau would end as soon as it found the contradiction, while our tableau would always take longer to reason about $p \wedge \neg p \wedge \phi$ than ϕ. Although not discussed in this paper, it is known how to implement both the CTL and BCTL* tableaux in a traditional rooted way [5], [10]. We would recommend converting this tableau to a conventional tableau prior to implementation.

Continued research into pure tableaux is important. [3] note that their hybrid implementation tends to perform better than that of [4]; however, they start with a rooted tableau it is not clear whether this is due to their approach or because they begin with conventional tableaux. Converting the approach of Reynolds to a conventional tableau, that begins with a single formula, greatly increases the performance a related tableau for BCTL* [10]. This suggests than even when performance is the only concern, research into pure-tableau is still worthwhile. As pure-tableaux based algorithms work directly on subformulae of the input formula, they have an important advantage over the hybrid technique of [3]: the workings of the algorithm is more easily understood by the user than a parity game solver.

There has been research into parallelisation of automated reasoning for CTL. For example, [13] propose a tableau for CTL that is intended to provide good average case performance and is easy to parallelise. The current CTL* tableaux do not yet exploit parallelisation. Single core performance of modern CPUs has plateaued. To exploit future advances in computation power, we will examine the potential to parallelise these tableau based techniques.

References

[1] E. A. Emerson and A. P. Sistla, "Deciding branching time logic," in *Proceedings of the 16th annual ACM symposium on Theory on computing (STOC)*. New York, NY, USA: ACM Press, 1984, pp. 14–24.

[2] M. Y. Vardi and L. Stockmeyer, "Improved upper and lower bounds for modal logics of programs," in *Proceedings of the 17th annual ACM symposium on Theory of computing (STOC)*. New York, NY, USA: ACM, 1985, pp. 240–251.

[3] O. Friedmann, M. Latte, and M. Lange, "A decision procedure for CTL* based on tableaux and automata," in *5th International Joint Conference on Automated Reasoning (IJCAR)*, ser. LNCS, J. Giesl and R. Hähnle, Eds. Springer, 2010, vol. 6173, pp. 331–345. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14203-1_28

[4] M. Reynolds, "A tableau for CTL*," in *Proceedings of the 16th International Symposium on Formal Methods (FM)*, ser. Lecture Notes in Computer Science, A. Cavalcanti and D. Dams, Eds., vol. 5850. Springer, 2009, pp. 403–418.

[5] E. A. Emerson and J. Y. Halpern, "Decision procedures and expressiveness in the temporal logic of branching time," in *STOC*. ACM, 1982, pp. 169–180. [Online]. Available: http://dx.doi.org/10.1145/800070.802190

[6] M. Reynolds, "A Tableau for Bundled CTL*," *J Logic Computation*, vol. 17, no. 1, pp. 117–132, 2007. [Online]. Available: http://logcom.oxfordjournals.org/cgi/content/abstract/17/1/117

[7] A. Masini, L. Viganò, and M. Volpe, "A labeled natural deduction system for a fragment of CTL*," in *Proceedings of the 2009 International Symposium on Logical Foundations of Computer Science*, ser. LFCS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 338–353. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-92687-0_23

[8] E. A. Emerson and C.-L. Lei, "Modalities for model checking (extended abstract): branching time strikes back," in *POPL '85: Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. New York, NY, USA: ACM, 1985, pp. 84–96.

[9] O. Kupferman and O. Grumberg, "Buy one, get one free!!!" *J. Log. Comput.*, vol. 6, no. 4, pp. 523–539, 1996. [Online]. Available: http://logcom.oxfordjournals.org/content/6/4/523.full.pdf

[10] J. C. M^{c}Cabe-Dansted, "A rooted tableau for BCTL*," 2011, Expanded Version, Availiable: http://www.csse.uwa.edu.au/~john/papers/Rooted_BCTL_Tableau.pdf.

[11] ——, "A temporal logic of robustness," Ph.D. dissertation, The University of Western Australia, 2011. [Online]. Available: http://tinyurl.com/RoCTL11

[12] ——, "Improved BCTL* applet," 2011, http://www.csse.uwa.edu.au/~john/BCTL2/.

[13] P. Abate, R. Goré, and F. Widmann, "One-pass tableaux for computation tree logic," in *Logic for Programming, Artificial Intelligence, and Reasoning*. Springer, 2007, pp. 32–46.