A Faster Tableau for CTL*

Mark Reynolds
School of Computer Science and Software Engineering, The University of Western Australia
mark.reynolds@uwa.edu.au

There have been several recent suggestions for tableau systems for deciding satisfiability in the practically important branching time temporal logic known as CTL*. In this paper we present a streamlined and more traditional tableau approach built upon the author's earlier theoretical work.

Soundness and completeness results are proved. A prototype implementation demonstrates the significantly improved performance of the new approach on a range of test formulas. We also see that it compares favourably to state of the art, game and automata based decision procedures.

1 Introduction

CTL* [5, 3] is an expressive branching-time temporal logic extending the standard linear PLTL [13]. The main uses of CTL* are for developing and checking the correctness of complex reactive systems [6] and as a basis for languages (like ATL*) for reasoning about multi-agent systems [8].

Validity of formulas of CTL* is known to be decidable with an automata-based decision procedure of deterministic double exponential time complexity [5, 4, 18]. There is also an axiomatization [14]. Long term interest in developing a tableau approach as well has been because they are often more suitable for automated reasoning, can quickly build models of satisfiable formulas and are more human-readable. Tableau-style elements have indeed appeared earlier in some model-checking tools for CTL* but tableau-based satisfiability decision procedures have only just started to be developed [17, 7].

Our CTL* tableau is of the tree, or top-down, form. To decide the validity of ϕ, we build a tree labelled with finite sets of sets of formulas using ideas called hues and colours originally from [14] and further developed in [16, 17]. The formulas in the labels come from a closure set containing only subformulas of the formula being decided, and their negations. Those earlier works proposed a tableau in the form of a roughly tree-shaped Hintikka-structure, that is, it utilised labels on nodes which were built from maximally consistent subsets of the closure set. Each formula or its negation had to be in each hue. In this paper we make the whole system much more efficient by showing how we only need to consider subformulas which are relevant to the decision.

In the older papers we identified two sorts of looping: good looping allowed up-links in our tableau tree while bad looping showed that a branch was just getting longer and longer in an indefinite way. In this paper we tackle only the good looping aspect and leave bad looping for a follow-on paper.

A publicly available prototype implementation of the approach here is available and comparisons with existing state of the art systems, and its Hintikka-style predecessor, show that we are achieving orders of magnitude speed-ups across a range of examples. As with any other pure tableau system, though, this one is better at deciding satisfiable formulas rather than unsatisfiable ones.

In section 2 we give a formal definition of CTL* before section 3 defines some basic building block concepts. Subsequent sections introduce the tableau shape, contain an example, look at a loop checking rule and show soundness. Section 7 presents the tableau construction rules and then we show completeness. Complexity, implementation and comparison issues are discussed briefly in section 10 before a conclusion. There is a longer version of this paper available as [15].

Gabriele Puppis, Tiziano Villa (Eds.): Fourth International
Symposium on Games, Automata, Logics and Formal Verification
EPTCS 119, 2013, pp. 50–63, doi:10.4204/EPTCS.119.7

2 Syntax and Sematics

Fix a countable set $\mathscr{L}$ of atomic propositions. A (transition) structure is a triple $M = (S, R, g)$ where:

S	is the non-empty set of *states*
R	is a total binary relation $\subseteq S \times S$ i.e. for every $s \in S$, there is some $t \in S$ such that $(s,t) \in R$.
g	$: S \to \mathscr{P}(\mathscr{L})$ is a labelling of the states with sets of atoms.

Formulas are defined along ω-long sequences of states. A *fullpath* in (S, R) is an infinite sequence $\langle s_0, s_1, s_2, \ldots \rangle$ of states such that for each i, $(s_i, s_{i+1}) \in R$. For the fullpath $\sigma = \langle s_0, s_1, s_2, \ldots \rangle$, and any $i \geq 0$, we write σ_i for the state s_i and $\sigma_{\geq i}$ for the fullpath $\langle s_i, s_{i+1}, s_{i+2}, \ldots \rangle$.

The formulas of CTL* are built from the atomic propositions in $\mathscr{L}$ recursively using classical connectives $\neg$ and $\wedge$ as well as the temporal connectives X, U and A. We use the standard abbreviations, **true**, **false**, $\vee$, $\to$, $\leftrightarrow$, $F\alpha \equiv \mathbf{true}\, U\alpha$, $G\alpha \equiv \neg F \neg \alpha$, and $E\alpha \equiv \neg A \neg \alpha$.

Truth of formulas is evaluated at fullpaths in structures. We write $M, \sigma \models \alpha$ iff the formula α is true of the fullpath σ in the structure $M = (S, R, g)$. This is defined recursively by:

$M, \sigma \models p$	iff	$p \in g(\sigma_0)$, any $p \in \mathscr{L}$
$M, \sigma \models \neg\alpha$	iff	$M, \sigma \not\models \alpha$
$M, \sigma \models \alpha \wedge \beta$	iff	$M, \sigma \models \alpha$ and $M, \sigma \models \beta$
$M, \sigma \models X\alpha$	iff	$M, \sigma_{\geq 1} \models \alpha$
$M, \sigma \models \alpha U \beta$	iff	there is $i \geq 0$ such that $M, \sigma_{\geq i} \models \beta$ and for each j, if $0 \leq j < i$ then $M, \sigma_{\geq j} \models \alpha$
$M, \sigma \models A\alpha$	iff	for all fullpaths σ' such that $\sigma_0 = \sigma'_0$ we have $M, \sigma' \models \alpha$

We say that α is *valid* in CTL*, iff for all transition structures M, for all fullpaths σ in M, we have $M, \sigma \models \alpha$. Say α is *satisfiable* in CTL* iff for some transition structure M and for some fullpath σ in M, we have $M, \sigma \models \alpha$. Clearly α is satisfiable iff $\neg\alpha$ is not valid.

3 Hues, Colours and Hintikka Structures

Fix the formula ϕ whose satisfiability we are interested in. We write $\psi \leq \phi$ if ψ is a subformula of ϕ. The length of ϕ is $|\phi|$. The *closure set* for ϕ is $\mathbf{cl}\,\phi = \{\psi, \neg\psi \mid \psi \leq \phi\}$.

Definition. [MPC] Say that $a \subseteq \mathbf{cl}\,\phi$ is *maximally propositionally consistent (MPC)* for ϕ iff for all $\alpha, \beta \in \mathbf{cl}\,\phi$, M1) if $\beta = \neg\alpha$ then ($\beta \in a$ iff $\alpha \notin a$); and M2) if $\alpha \wedge \beta \in \mathbf{cl}\,\phi$ then ($\alpha \wedge \beta \in a$ iff both $\alpha \in a$ and $\beta \in a$).

The concepts of hues and colours were originally invented in [14] but we use particular formal definitions as presented in [16, 17, 15]. A hue is supposed to capture (approximately) a set of formulas which could all hold together of one fullpath. Definition. [hue] $a \subseteq \mathbf{cl}\,\phi$ is a *hue* for ϕ, or ϕ-hue, iff all these conditions hold:

H1) a is MPC;
H2) if $\alpha U \beta \in a$ and $\beta \notin a$ then $\alpha \in a$;
H3) if $\alpha U \beta \in (\mathbf{cl}\,\phi) \setminus a$ then $\beta \notin a$;
H4) if $A\alpha \in a$ then $\alpha \in a$.

Further, let H_ϕ be the set of hues of ϕ.

For example, if $\neg(AG(p \to EXp) \to (p \to EGp))$, the example known as $\neg\theta_{12}$ in [17], then here is a hue known as $h38$:

$$\{\neg(AG(p \to EXp) \to (p \to EGp)), (AG(p \to EXp) \wedge \neg(p \to EGp)),$$
$$AG(p \to EXp), G(p \to EXp), \mathbf{true}, \neg\neg(p \to EXp),$$
$$(p \to EXp), p, \neg\neg EXp, EXp, \neg\neg Xp, Xp,$$
$$\neg(p \to EGp), (p \wedge \neg EGp), \neg EGp, A\neg Gp, \neg Gp, F\neg p, \neg\neg p\}$$

The usual temporal successor relation plays a role in determining allowed steps in the tableau. The relation r_X is put between hues a and b if a fullpath σ satisfying a could have a one-step suffix $\sigma_{\geq 1}$ satisfying b: Definition. [r_X] For hues a and b, put $a\, r_X\, b$ iff the following four conditions all hold:

R1) if $X\alpha \in a$ then $\alpha \in b$;
R2) if $\neg X\alpha \in a$ then $\neg\alpha \in b$;
R3) if $\alpha U\beta \in a$ and $\neg\beta \in a$ then $\alpha U\beta \in b$; and
R4) if $\neg(\alpha U\beta) \in a$ and $\alpha \in a$ then $\neg(\alpha U\beta) \in b$.

We also introduced an equivalence relation aiming to tell whether two hues could correspond to fullpaths starting at the same state. We just need the hues to agree on atoms and on universal path quantified formulas: Definition. [r_A] For hues a and b, put $a\, r_A\, b$ iff the following two conditions both hold: A1) for all $p \in \mathscr{L}$, $p \in a$ iff $p \in b$; and A2) $A\alpha \in a$ iff $A\alpha \in b$.

Now we move up from the level of hues to the level of colours. Could a set of hues be exactly the hues corresponding to all the fullpaths starting at a particular state? We would need each pair of hues to satisfy r_A but we would also need hues to be in the set to witness all the existential path quantifications:

Definition. [colour] Non-empty $c \subseteq H_\phi$ is a *colour* of ϕ, or ϕ-colour, iff the following two conditions hold. For all $a, b \in c$, C1) $a\, r_A\, b$; and C2) if $a \in c$ and $\neg A\alpha \in a$ then there is $b \in c$ such that $\neg\alpha \in b$. Let C_ϕ be the set of colours of ϕ.

The formulas $\neg Xp, EXp$ are both in $h37$, another hue from the example in [17], so $\{h37\}$ is not a colour. However, $Xp \in h38$ witnesses the existential path quantification so $\{h37, h38\}$ is a colour.

We define a successor relation R_X between colours. It is defined in terms of the successor relation r_X between the component hues and it will be used to define the successor relation between tableau nodes, themselves corresponding to states in transition structures, in terms of the colours which they exhibit. Note that colours, and tableau nodes, will, in general, have a non-singleton range of successors and this relation R_X just tells us whether one node can be one of the successors of another node.

Definition. [R_X] For all $c, d \in C_\phi$, put $c\, R_X\, d$ iff for all $b \in d$ there is $a \in c$ such that $a\, r_X\, b$.

It is worth noting that colours and hues are induced by actual transition structures. We will need these concepts in our completeness proof.

Definition. [actual ϕ-hue] Suppose (S, R, g) is a transition structure. If σ is a fullpath through (S, R) then we say that $h = \{\alpha \in \mathbf{cl}\,\phi \mid (S, R, g), \sigma \models \alpha\}$ is the *actual (ϕ-) hue* of σ in (S, R, g).

It is straightforward to see that this is a ϕ-hue. It is also easy to show that along any fullpath σ, the relation r_X holds between the actual hue of σ and the actual hue of its successor fullpath $\sigma_{\geq 1}$.

Definition. [actual ϕ-colour] If $s \in S$ then the set of all actual hues of all fullpaths through (S, R) starting at s is called the *actual (ϕ-) colour* of s in (S, R, g).

Again, it is straightforward to show that this is indeed a ϕ-colour and also that R_X holds between the actual colour of any state and the actual colour of any of its successors.

4 Tableau

The tableaux we construct will be roughly tree-shaped: the traditional upside down tree with a root at the top, predecessors and ancestors above, successors and descendants below. However, we will allow

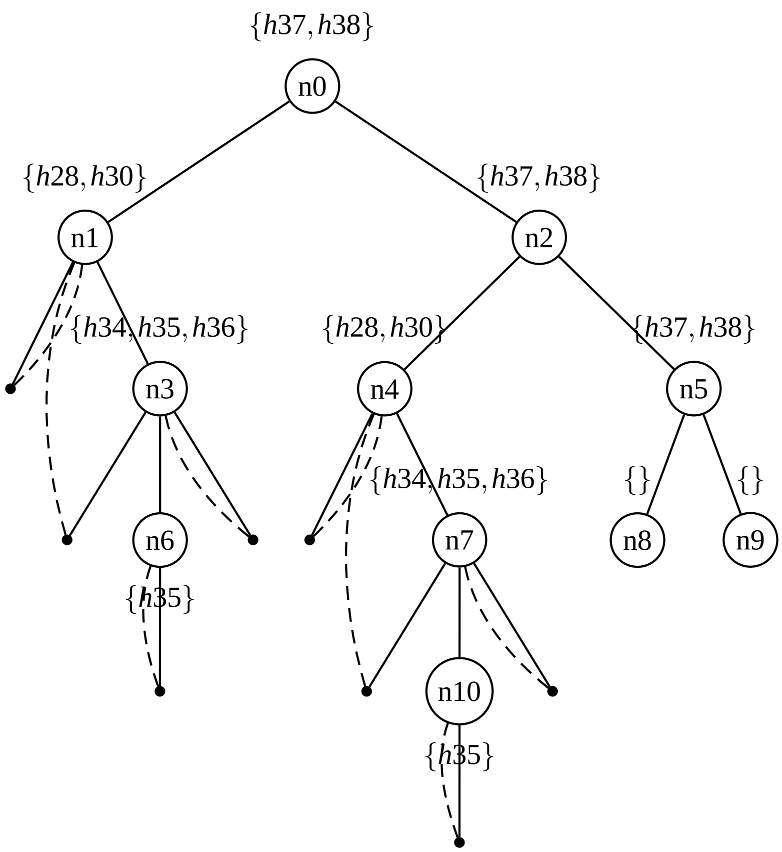

Figure 1: A Partial Tableau for $\neg\theta_{12}$

Definition. A *tableau* for $\phi \in L$ is a tuple (T, s, η, π) such that:

H1) T is a non-empty set of *nodes*; one distinguished element called the *root*;

H2) η is the phue label enumerator, so that for each $t \in T$, $\eta_t : \mathbb{N} \to 2^{\mathbf{cl}\,\phi}$ is a partial map,

H2.1) the domain of η_t is $\{0, 1, ..., n-1\}$ for some $n > 0$ denoted $|\eta_t|$;

H2.2) $\eta_t(i)$ is the ith label phue of t (if defined);

H3) s is the successor enumerator, so that for each $t \in T$, $s_t : \mathbb{N} \to T$ is a partial map,

H3.1) the domain of s_t is a subset of $\{0, 1, ..., |\eta_t| - 1\}$; $s_t(i)$ the ith successor of t;

H3.3) for each $t \in T$, there is a unique finite sequence $r_0, r_1, ..., r_k$ from T called the *ancestors* of t such that the r_i are all distinct, r_0 is the root, $r_k = t$ and for each j, r_{j+1} is a successor of r_j;

H4) $\phi \in \eta_{\mathbf{root}}(0)$;

H5) π is the predecessor map whereby if $t, u \in T$ then either π^t_u is undefined and we say that t is not a predecessor of u; or for all $j < |u|$, $\pi^t_u(j) = i < |t|$ and we say that the ith phue in t is a predecessor of the j th hue in u.

H6) if $s_t(i) = u$ then $\pi^t_u(0) = i$ (i.e. the ith phue in t is a predecessor of the 0th phue in $s_t(i)$);

Figure 2: Definition of Tableau

up-links from a node to one of its ancestors. Each node will be labelled with a finite sequence of sets of formulas from the closure set. We will call such a sequence of sets a *proto-colour* or *pcolour*. The sets, or *proto-hues (phues)*, in the pcolour are ordered and once completed the node will have one (ordered) successor for each phue.

The ordering of the successors will match the ordering of the hues (H3.1 and H6) so that we know there is a successor node containing a successor phue for each phue in the label. The respective orderings are otherwise arbitrary.

A *proto-hue (phue)* is just a subset of $\mathbf{cl}\,\phi$.

See Figure 2 for our definition of a tableau.

Definition. Say that the tableau (T, s, η, π) has *supported labelling* if each formula in each phue in each label is supported, as follows. Consider a formula $\alpha \in \eta_t(i)$. Determining whether α is support for not depends on the form of α:

- p is supported in $\eta_t(0)$. Otherwise, i.e. for $i > 0$, it is only supported if $p \in \eta_t(0)$.
- Same with $\neg p$.
- $\neg\neg\alpha$ supported iff $\alpha \in \eta_t(i)$.
- $\alpha \wedge \beta$ supported iff $\alpha \in \eta_t(i)$ and $\beta \in \eta_t(i)$.
- $\neg(\alpha \wedge \beta)$ supported iff either $\neg\alpha \in \eta_t(i)$ or $\neg\beta \in \eta_t(i)$.
- $X\alpha \in \eta_t(i)$ supported iff 1) there is $u \in T$ with $u = s_t(i)$ and 2) for all $u \in T$, for all j with $\pi^t_u(j) = i$, $\alpha \in \eta_u(j)$.
- $\neg X\alpha \in \eta_t(i)$ supported iff 1) there is $u \in T$ with $u = s_t(i)$ and 2) for all $u \in T$, for all j with $\pi^t_u(j) = i$, $\neg\alpha \in \eta_u(j)$.
- $\alpha U\beta \in \eta_t(i)$ supported iff 1) $\beta \in \eta_t(i)$; or 2) all 2.1) $\alpha \in \eta_t(i)$; 2.2) there is $u \in T$ with $u = s_t(i)$; and 2.3) for all $u \in T$, for all j with $\pi^t_u(j) = i$, $\alpha U\beta \in \eta_u(j)$.
- $\neg(\alpha U\beta) \in \eta_t(i)$ supported iff 1) $\neg\beta \in \eta_t(i)$; and 2) either 2.1) $\neg\alpha \in \eta_t(i)$; or 2.2) both 2.2.1) there is $u \in T$ with $u = s_t(i)$; and 2.2.2) for all $u \in T$, for all j with $\pi^t_u(j) = i$, $\neg(\alpha U\beta) \in \eta_u(j)$.
- $A\alpha \in \eta_t(i)$ supported iff for all $j < |\eta_t|$, $\alpha \in \eta_t(j)$.
- $\neg A\alpha \in \eta_t(i)$ supported iff there is some $j < |\eta_t|$, $\neg\alpha \in \eta_t(j)$.

A tableau is *successfully finished* iff it has no leaves, the predecessor relation is defined on all phues and the tableau does not fail any of the three checks that we introduce below: LG, NTP and the non-

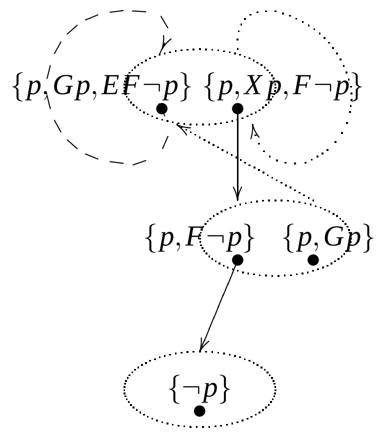

Figure 3: Example tableau.

existence of direct contradictions (or **false**) in phues.

It is common, in proving properties of tableau-theoretic approaches to reasoning, to refer to labelled structures as *Hintikka structures* if the labels are maximally complete (relative to a closure set). We say that one of our tableaux (T, s, η, π) is a Hintikka tableau iff the elements of each η_t are all hues (not just any phues). The older tableau approach in [17] was based on Hintikka tableaux.

5 Tableau Examples

Figure 1 is an example (unfinished) tableau illustrating general shape. There are 11 nodes, each with successors marked, and each labeled with a set of phues. Note that some of the successor relations involve up-links: $n1$ is a successor of $n3$. We just name the phues rather than listing their contents. There are more details about this example in [17] as, in fact, it is a Hintikka-tableau, which is a special type of the tableau we are introducing in this paper. We use Hintikka-tableaux later in the completeness proof here.

Figure 3 shows a smaller tableau in more detail. He we show the phues, which make up the pcolour labels of nodes and we show the predecessor or traceback map in some cases.

6 The LG test and Soundness

In this section we will briefly describe the LG rule which is a tableau construction rule that prevents bad up-links being added. LG is used to test and possibly fail a tableau. The test is designed to be used soon after any new up-link is added after being proposed by the LOOP rule. If the new tableau fails the LG test then "undo" the up-link and continue with alternative choices. We then show that if a tableau finishes, that is has no leaves, and passes the LG test then it guarantees satisfiability.

There was also a very similar LG test in the earlier work on the original slower tableau method [17]. In that paper, we show how to carry out the LG check on a tableau and we prove some results about its use. The check is very much like a model check on the tableau so far. We make sure that every phue in a label *matches*, or is a subset of an actual hue at that node in a transition structure defined using a

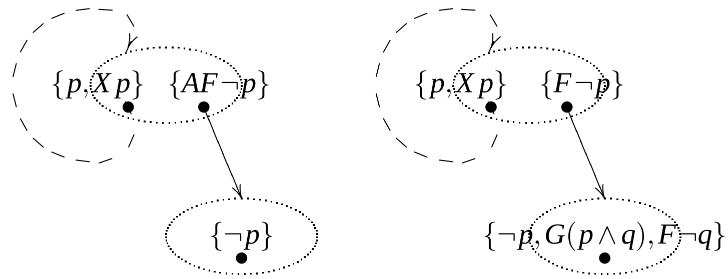

Figure 4: LG examples: left fails LG; right passes but eventually does not succeed

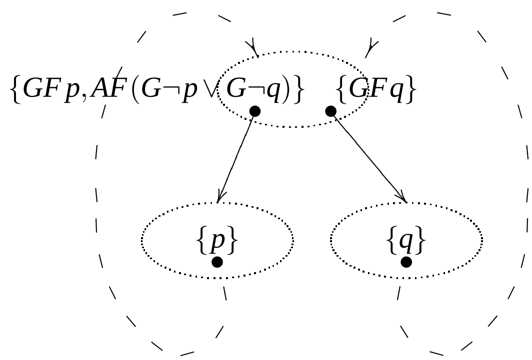

Figure 5: These two loops fail LG.

valuation of atoms based on the labels. It has polynomial running time in the size of the tableau so it is not a significant overhead on the overall tableau construction algorithm.

Due to space restrictions we do not go through the full details of the only very slightly different LG rule used for the faster tableaux here. Instead we give some brief motivation examples. The first example shows us that not all up-links are allowable: e.g., a node labelled with $p, AF\neg p$ which also has an immediate loop. See left hand example in Figure 4. The up-link would not be allowed by the LG rule.

The right hand example in Figure 4, with an allowable up-link and also separately an unsatisfiable leaf, is allowed by LG.

The example in Figure 5 has two loops, each one individually acceptable but not both. The LG rule fails the tableau when both up-links are added.

Now we show that if ϕ has a successfully finished tableau then ϕ is satisfiable. This is the soundness Lemma.

Lemma. If ϕ has a successfully finished tableau then ϕ is satisfiable.

Here we just outline the proof: details in [15]. Say that (T, s, η, π) is a successfully finished tableau for ϕ. Define a structure $M = (T, R, g)$ by interpreting the s relation as a transition relation g, and using η to define the valuation g on nodes.

By definition of matching, after a final check of LG there is some actual hue b of the root such that $\eta_{\mathbf{root}}(0) \subseteq b$. This means that ϕ holds along some fullpath in the final structure.

7 Building a tree

In this section we briefly describe how a tableau is built via some simple operations, or rules. We start with an initial tree of one root node labelled with just one phue containing only ϕ. The rules allow formulas to be added inside hues in labels, new hues to be added in labels and new nodes to be added as successors of existing nodes. The rules are generally non-deterministic allowing a finite range of options, or choices, at any application.

There are some properties to check such as LG, described above, and NTP described below. We also check that there are no hues containing both a formula and its negation, and we check that **false** is not contained in a phue. If these checks fail then the tableau has failed and we will need to backtrack to explore other possible options at choice points along the way.

The tableau succeeds if there are no leaves.

7.1 Basic Tableau Rules

Here are most of the basic rules, in an abbreviated notation:

2NEG: $\frac{\{\{\neg\neg\alpha\}\}}{\{\{\alpha\}\}}$ CONJ: $\frac{\{\{\alpha\wedge\beta\}\}}{\{\{\alpha,\beta\}\}}$ DIS: $\frac{\{\{\neg(\alpha\wedge\beta)\}\}}{\{\{\neg\alpha\}\}\mid\{\{\neg\beta\}\}}$ NEX: $\frac{\{\{X\alpha\}\}\to\{\{\}\}}{\{\{X\alpha\}\}\to\{\{\alpha\}\}}$ NNX: $\frac{\{\{\neg X\alpha\}\}\to\{\{\}\}}{\{\{\neg X\alpha\}\}\to\{\{\neg\alpha\}\}}$

UNT: $\frac{\{\{\alpha U\beta\}\}\to\{\{\}\}}{\{\{\alpha U\beta,\beta\}\}\to\{\}\mid\{\{\alpha U\beta,\alpha\}\}\to\{\{\alpha U\beta\}\}}$ NUN: $\frac{\{\{\neg(\alpha U\beta)\}\}\to\{\{\}\}}{\{\{\neg(\alpha U\beta),\neg\beta,\neg\alpha\}\}\to\{\}\mid\{\{\neg(\alpha U\beta),\neg\beta,\alpha\}\}\to\{\{\neg(\alpha U\beta)\}\}}$

ATM: $\frac{\{\{p\},\{\}\}}{\{\{p\},\{p\}\}}$ NAT: $\frac{\{\{\neg p\},\{\}\}}{\{\{\neg p\},\{\neg p\}\}}$ POS: $\frac{\{\{\neg A\alpha\}\}}{\{\{\neg A\alpha,\neg\alpha\}\}\mid\{\{\neg A\alpha\},\{\neg\alpha\}\}}$ NEC: $\frac{\{\{A\alpha\},\{\}\}}{\{\{A\alpha,\alpha\},\{\alpha\}\}}$

The rules are described in detail in [15] but the notation gives the main ideas. Here are details of a few of the rules above.

DIS: If $\neg(\alpha\wedge\beta)\in\eta_t(j)$ then can extend (T,s,η,π) to (T',s',η',π') via either: DIS1 or DIS2 as follows. DIS1 produces (T',s',η',π') such that $T'=T$, $s'=s$, and for all $t'\neq t$, $\eta_{t'}=\eta_t$ and for all $i'\neq i$, $\eta'_t(i')=\eta_t(i')$. However, $\eta'_t(i)=\eta_t(i)\cup\{\neg\alpha\}$. DIS2 is similar but use β instead of α.

NEX: If $X\alpha\in\eta_t(i)$ and there is $u\in T$ and j with $\pi^t_u(j)=i$ then can extend (T,s,η,π) to (T',s',η',π') such that $T'=T$, $s'=s$, and $\eta'_u(j)=\eta_u(j)\cup\{\alpha\}$. If $t\in T$ but there is no $s_t(j)\in T$ then extend (T,s,η,π) to (T',s',η',π') using new object t^+ such that $T'=T\cup\{t^+\}$, $s'_t(i)=t^+$, $\eta'_{t^+}(0)=\{\}$ and $\pi'^t_{t^+}(0)=i$. For all other arguments, s', η' and π' inherit values from s,η and π respectively.

ATM: If an atom $p\in\eta_t(j)$ and $k<|\eta_t|$ then can extend (T,s,η,π) to (T',s',η',π') such that $T'=T$, $s'=s$, and for all $t'\neq t$, $\eta_{t'}=\eta_t$ and for all $i'\neq k$, $\eta'_t(i')=\eta_t(i')$. However, $\eta'_t(k)=\eta_t(k)\cup\{p\}$.

POS: If $\neg A\alpha\in\eta_t(j)$ and $n=|\eta_t|$ then can extend (T,s,η,π) to (T',s',η',π') via one of POS_k for some $k=0,1,2,...,n$ as follows. For $k<n$, POS_k involves extending (T,s,η,π) to (T',s',η',π') where $T'=T$, $s'=s$, and for all $t'\neq t$, $\eta_{t'}=\eta_t$ and for all $i'\neq k$, $\eta'_t(i')=\eta_t(i')$. However, $\eta'_t(k)=\eta_t(k)\cup\{\neg\alpha\}$. However, POS_n involves extending (T,s,η,π) to (T',s',η',π') where $T'=T$, $s'=s$, and for all $t'\neq t$, $\eta_{t'}=\eta_t$ and for all $i'\neq k$, $\eta'_t(i')=\eta_t(i')$. However, $\eta'_t(k)=\eta_t(k)\cup\{\neg\alpha\}$.

There are also a couple of rules not sketched above.

PRED: If $t,u\in T$ and u is a successor of t but $\pi(t_u(j))$ is not defined then we can extend (T,s,η,π) to (T',s',η',π') via one of PRED_k for some $k=0,1,2,...,|\eta_t|-1$ as follows.

For $k<|\eta_t|$, PRED_k involves extending (T,s,η,π) to (T',s',η',π') where $T'=T$, $s'=s$, and $\eta'=\eta$. However, $\pi'^t_u(j)=k$.

For $k = |\eta_t|$, PRED$_k$ involves extending (T,s,η,π) to (T',s',η',π') where $T' = T$, but $\eta' = \eta$ but giving t an extra empty phue $\eta'_t(k) = \{\}$; and $s = s'$.

Later we need to add a kth successor for t and fill in formulas in $\eta'_t(k)$.

Note that t now potentially becomes unsupported, untraceable and unfinished, again.

LOOP: Suppose t is an ancestor of the parent u^- of u, then we can choose either to replace the u^- to u edge by an up-link from u^- to t, or to not do that replacement (and continue the branch normally).

(It is worth remembering which choice you make and not try that again if it did not work.)

Note that, as in normal successors, we will also put $s_{u^-}(i) = t$ and $\pi_t^{u^-}(0) = i$ where previously we had $s_{u^-}(i) = u$. All the other phues in η_t will also have to have predecessors chosen amongst the phues in η_{u^-}. We will use the PRED rule to do this for each one.

Note also that making such an up-link can possibly cause a subsequent consequential failure of the tableau. A contradiction could be introduced into the hues of t, the NTP could fail and/or the LG property could fail. It is possible to test for a few of these potential problems just before making use of this rule and act accordingly.

7.2 The NTP check: nominated thread property

The LG property check that every looping path is noticed by the labels in nodes. The converse requirement is taken care of by the much simpler NTP check.

We put a special significance on the initial hue in each colour label. This, along with the next condition, helps us ensure that each hue actually has a fullpath witnessing it. We are going to require the following property, NTP, of the tableaux which we construct.

First some auxiliary definitions: Definition. [hue thread] Suppose σ is a path through (T,s,η,π). A *hue thread* through σ is a sequence ξ of hues such that $|\xi| = |\sigma|$, for each $j < |\xi|$, $\xi_j \in \eta(\sigma_j)$ and for each $j < |\xi| - 1$, $\xi_j r_X \xi_{j+1}$.

Definition. [fulfilling hue thread] Suppose σ is a path through (T,s,η,π) and ξ is a hue thread through σ. We say that ξ is fulfilling iff either $|\sigma| < \omega$, or $|\sigma| = \omega$ and all the eventualities in each ξ_i are witnessed by some later ξ_j; i.e. if $\alpha\, U \beta \in \xi_i$ then there is $j \geq i$ such that $\beta \in \xi_j$.

Definition. [the nominated thread property] We say that the tableau (T,s,η,π) has the *nominated thread property* (NTP) iff the following holds. Suppose that for all $t \in T$ such that $0 < |s_t|$, $s_t(0)$ is an ancestor of t and that $t_0 = s_t(0), t_1, \ldots, t_k = t$ is a non-repeating sequence with each $t_{j+1} = s_{t_j}(0)$. Let σ be the fullpath $\langle t_0, t_1, \ldots, t_k, t_0, t_1, \ldots, t_k, t_0, t_1, \ldots \rangle$ and ξ be the sequence $\langle \eta_{t_0}(0), \eta_{t_1}(0), \ldots, \eta_{t_k}(0), \eta_{t_0}(0), \ldots \rangle$ of hues in σ. Then ξ is a fulfilling hue thread for σ.

It is straightforward to prove that this is equivalent to checking that each eventuality in $\eta_{t_0}(0)$ (or in all, or any, $\eta_{t_i}(0)$) is witnessed in at least one of the $\eta_{t_j}(0)$. So it is neither hard to implement nor computationally complex.

Using the rules described above, using any applicable one at any stage, allows construction of tableaux. We know that the LG rule ensures that any successful ones which we build thus will guarantee that ϕ is satisfiable. In the next section we consider whether we can build a successful tableau for any satisfiable formula in the way.

8 Completeness Using the Hintikka Tableau

In [17], the completeness result for the tableau in that paper, shows that for any satisfiable CTL* formula there is a finite model satisfying certain useful properties and from that we can find a successful tableau (as defined in that paper) for the formula. In fact the tableau constructed in that paper is just a special form of the tableaux that we are constructing in this paper: they are Hintikka structures.

Definition. A structure (T, s, η, π) is a *Standard Hintikka Tableau* for ϕ iff (T, s, η, π) is a finite finished successful tableau for ϕ and for each t, for each i, $\eta_t(i)$ is an MPC subset of $\mathbf{cl}\,(\phi)$.

Thus, in a Hintikka tableau, the labels tell us exactly which formulas hold there.

The completeness result in [17] shows the following, in terms of the concepts defined in this paper:

Lemma. If $\phi \in L$ is satisfiable then it has a Standard Hintikka Tableau.

The proof of this lemma is a straightforward translation of the definitions from [17] but we need to specify how to define our current predecessor relation π and we also need to check that the tableau is finished.

The predecessor relation π is not made explicit in the tableau structures of the earlier paper. Instead we require that the colour of a node t is related by a successor relation R_X between colours to the colour of any successor t'. This means that for any hue in the colour of t' there is a hue h in the colour of t such that h and h' are related by a successor relation between hues. We can use such a hue h as the predecessor of h' and so define π.

To show that the tableau (T, s, η, π) is finished, we just need to check all the rules of our tableau construction and make sure none require the tableau to be changed in any way. This needs to be done each rule at a time, and needs to be done carefully, although it is straightforward.

The proof in [17] uses a finite model theorem for CTL* to obtain a *branch boundedness* result on the Hintikka tableau. We can guarantee existence of a such a tableau with a certain function of the length of the formula bounding the length of each branch (before an up-link). The bound is triple exponential in the length of the formula, so rather large.

Thus we can conclude that each satisfiable formula has a tableau, but we can not yet claim that it is a tableau which can be constructed by our rules.

In the rest of this section we describe how we can show that if ϕ is satisfiable then there is a sequence of applications of our tableau rules that allow the construction of a successful tableau for ϕ. Suppose ϕ is satisfiable. From the lemma above we know that there is a successful, branch-bounded, supported tableau $T^{-\infty} = (T', s', \eta', \pi')$ for ϕ.

In [15], we show how to build a related, successful tableau for ϕ in a step by step manner only using the construction rules from section 7.1. Thus we make a sequence $T^0, T^1, \ldots$ of tableaux each one using a construction step to get to the next.

In order to use $T^{-\infty}$ to guide us, we also construct a sequence of maps $w_0, w_1, w_2, \ldots$, each w_i relating the phues of the labels of the nodes of T^i to the hues of the labels of the nodes of $T^{-\infty}$.

Thus each w_i maps ordered pairs which are nodes paired with indices to other such pairs. Suppose that $T^i = (T, s, \eta, \pi)$ and $T^{-\infty} = (T', s', \eta', \pi')$. Say $t \in T^i$ and $j < |\eta_t|$. Then $w_i(t, j)$ will be defined: say that $w_i(t, j) = (u, k)$ for $u \in T'$. Then $k < |\eta'_u|$. The idea in this example is that w_i is associating the jth phue of t with the kth phue of u.

All the while during the construction we ensure that w_i maps each node in T^i to a node in $T^{-\infty}$ which has a superset label.

We also show that the constructed tableau does not fail at any stage if one of the check rules such as LG, NTP or the existence of direct contradictions in phues. This follows from the fact that the phues in its labels are subsets of the hues in the labels of the Hintikka tableau.

If T is finished (leafless), supported and all predecessors exist then we are done. If T is not supported then choose any formula α in any phue in the label of any node that is not supported. Depending on the form of α we apply one of the tableau rules to add some successor, or some phue and/or some formula(s) in a phue that will ensure that α is then supported. See [15] for details.

There are only a finite number of formulas that can be added in hues in labels in a finite structure which is a subset of $T^{-\infty}$. This guarantees that the process will eventually terminate.

Thus every satisfiable formula has a successful tableau which can be found via our set of rules.

In fact, we can go further and get an even better completeness result. We can show that each formula ϕ only has a finite number of tableaux which respect the branch bounds and a simple bound on branching factor. Furthermore, if there is a successful tableau then there will be one obeying these bounds. There are at most $2^{|\phi|}$ hues and so each node in a Hintikka tableau has at most $2^{|\phi|}$ successors: by the form of completeness proof we can enforce the same bound on our more general tableaux. As we also have a finite bound on the length of branches there are clearly only finitely many tableaux for any particular ϕ.

Lemma. Given ϕ, there are only a finite number of tableaux which respect the branch length bound and the branching degree bounds.

In this definition of tableau we have guaranteed termination of any tableau construction algorithm by putting a simple but excessive bound on the length of branches. This allows us to conclude failure in a finite time and to also abbreviate the search for successful tableaux.

9 Stopping Repetition: coming up in follow-on paper

In this paper we have only briefly mentioned the limit on the length of branches as a way of guaranteeing that there are only finitely many tableau, and so that a search will terminate one way or another. The limit, based on a theoretical upper bound on the minimal CTL* model size, is very generous and hence this is an inefficient way of cutting short tableau searches. Being so generous slows down both negative and positive satisfiability reports.

In order to make some sort of working implementation to demonstrate the practicality of this tableau it is necessary to have a better way of preventing the construction of wastefully long branches. For want of better terminology we will call such a facility, a "repetition checker".

The task of making a quick and more generally usable repetition checker will be left to be advanced and presented at a later date. In fact, eventually we hope to provide a useful set of criteria for earlier termination of construction of branches depending on the properties of the sequence of colours so far. A simple example of the sort of criterion is the repeated appearance of the same sequence of colours and hues along a non-branching path without being able to construct any up-links. Other more sophisticated ideas are easily suggested but we want to develop a more systematic set of tests before presenting this in future work.

In [17], we present some basic repetition checking tests for the Hintikka style tableau. These can be used in order to allow some faster automated tableau construction. The tests can be modified to work with our sparser labels, and we will present full details in a future paper. There are many opportunities for more thorough repetition checks as well.

10 Complexity, Implementation and Comparisons

Say that $|\phi| = l$. Thus ϕ has $\leq l$ subformulas and **cl** ϕ contains at most $2l$ formulas. Since each hue contains, for each $\alpha \leq \phi$ at most one of α or $\neg\alpha$, there are at most $\leq 2^l$ hues. Thus there are less than 2^{2^l} colours. It is straightforward to see that there is a triple exponential upper bound if the tableau search algorithm uses the double exponential bound on branch length [17] to curtail searches down long branches.

A prototype implementation written by the author shows that for many interesting, albeit relatively small, formulas, the experimental performance of the system is relatively impressive. There are some preliminary results detailed in [15] which show a comparison of running times with the older Hintikka-style tableau technique of [17] and the state of the art game-based CTL* reasoner from [7]. In general the new reasoner is more than an order of magnitude quicker at deciding formulas from a range of basic and distinctive CTL* validities and their negations and a few other satisfiable formulas. The implementation is available as Java code for public download [15]. Online reasoner coming soon.

The implementation for the new technique that is used in these experiments, uses some basic repetition checking derived from the checks given earlier in the Hintikka-style system [17]. The new, slightly modified versions of these mechanisms are not described in the current paper. Instead they will be described in a future paper.

In [7], four series of formulas are suggested to examine asymptotic behaviour. Timing results for our system on these formulas are presented in Table 6. We compare the performance of our new tableau with the state of the art in game-based techniques for deciding CTL*. This is using published performance of the reasoner from [7] as reported in experiments in [11]. Consider the following series of formulas: $\alpha_1 = AFGq$, $\beta_1 = AFAGq$ and for each $i \geq 1$, $\alpha_{i+1} = AFG\alpha_i$ and $\beta_{i+1} = AFAG\beta_i$. In table 6, we compare the performance of the Hintikka-style tableau system from [17], the game-based reasoner from [7] and the new tableau system of this paper (using basic repetition checking) on the growing series built from these formulas. Although the running times, are on different computers, and so not directly comparable, we can see the difference in asymptotic performance. Running times greater than an hour or two are curtailed. From the results we see that we have achieved very noticeable and significant improvements in performance on the satisfiable examples.

Pure tableau-style reasoning on unsatisfiable formulas often involves exhaustive searches and the new technique is not immune to such problems. See the 400 series of examples in the asymptotic experiments. We will say more about these examples when proposing some new repetition mechanisms in the future.

There are some, more theoretical descriptions of other game-based and automata-based techniques for model-checking CTL* in older papers such as [10], [2] and [9]. However, these do not seem directly applicable to satisfiability decisions and/or there do not seem to be any easily publicly available implemented tools based on these approaches.

11 Conclusion

In this paper we have presented, albeit in a fairly high level sketch, a traditional tableau approach to reasoning with the important logic CTL*. Soundness and completeness results are proved and prototype implementation demonstrates the significantly improved performance of the new approach on a range of test formulas.

The next task in this direction is to build on the foundation here and present full details and proofs of the repetition checking mechanisms that can be used with the tableau construction. There are some basic repetition mechanisms available in the previous, Hintikka style tableau [17] but they need to be modified slightly. There are opportunities for additional techniques. It is also important to improve and document the rule-choice algorithms which have a bearing on running times.

In the future, it will be useful to develop reasoning tools which combine the latest in tableaux, automata and game-based approaches to CTL*. Having tools working in parallel should allow faster decisions. It will also be useful to extend the work to logics of multi-agent systems such as ATL* and strategy logic [12].

#	formula	length	sat?	MRH [17]	FLL [7]	NEW this paper
101	$\alpha_1 \rightarrow \beta_1$	20	Y	330	120	39
102	$\alpha_2 \rightarrow \beta_2$	35	Y	$> 10^5$	130	43
103	$\alpha_3 \rightarrow \beta_3$	50	Y	out of time	120	69
108	$\alpha_8 \rightarrow \beta_8$	125	Y	out of time	380	664
113	$\alpha_{13} \rightarrow \beta_{13}$	200	Y	out of time	$> 10^5$	2677
115	$\alpha_{15} \rightarrow \beta_{15}$	230	Y	out of time	$> 10^6$	4228
119	$\alpha_{19} \rightarrow \beta_{19}$	290	Y	out of time	out of time	9468
201	$\neg(\alpha_1 \rightarrow \beta_1)$	21	Y	350	120	172
202	$\neg(\alpha_2 \rightarrow \beta_2)$	36	Y	$> 10^5$	170	117
203	$\neg(\alpha_3 \rightarrow \beta_3)$	51	Y	out of time	2270	213
204	$\neg(\alpha_4 \rightarrow \beta_4)$	66	Y	out of time	$> 10^6$	377
205	$\neg(\alpha_5 \rightarrow \beta_5)$	81	Y	out of time	out of time	673
212	$\neg(\alpha_{12} \rightarrow \beta_{12})$	186	Y	out of time	out of time	7153
301	$\beta_1 \rightarrow \alpha_1$	20	Y	340	130	48
302	$\beta_2 \rightarrow \alpha_2$	35	Y	$> 10^5$	140	50
303	$\beta_3 \rightarrow \alpha_3$	50	Y	out of time	140	86
312	$\beta_{12} \rightarrow \alpha_{12}$	185	Y	out of time	30970	3333
314	$\beta_{14} \rightarrow \alpha_{14}$	215	Y	out of time	$> 10^6$	5512
316	$\beta_{16} \rightarrow \alpha_{16}$	245	Y	out of time	out of time	8627
319	$\beta_{19} \rightarrow \alpha_{19}$	290	Y	out of time	out of time	15615
401	$\neg(\beta_1 \rightarrow \alpha_1)$	21	N	400	760	1801
402	$\neg(\beta_2 \rightarrow \alpha_2)$	36	N	$> 10^5$	48670	$> 10^5$
403	$\neg(\beta_3 \rightarrow \alpha_3)$	51	N	out of time	$> 10^6$	out of time

Figure 6: Asymptotic Examples: Running Times (milliseconds)

References

[1] Sergei N. Artëmov & Anil Nerode, editors (2009): *Logical Foundations of Computer Science, International Symposium, LFCS 2009, Deerfield Beach, FL, USA, January 3-6, 2009. Proceedings*. *Lecture Notes in Computer Science* 5407, Springer. Available at `http://dx.doi.org/10.1007/978-3-540-92687-0`.

[2] Orna Bernholtz, Moshe Y. Vardi & Pierre Wolper (1994): *An Automata-Theoretic Approach to Branching-Time Model Checking (Extended Abstract)*. In David L. Dill, editor: *CAV*, *Lecture Notes in Computer Science* 818, Springer, pp. 142–155. Available at `http://dx.doi.org/10.1007/3-540-58179-0_50`.

[3] E. Emerson & J. Halpern (1986): *'Sometimes' and 'not never' revisited*. *J. ACM* 33. Available at `http://dx.doi.org/10.1145/4904.4999`.

[4] E. Emerson & C. Jutla (1988): *Complexity of Tree Automata and Modal Logics of Programs*. In: *29th IEEE Foundations of Computer Science, Proceedings*, IEEE.

[5] E. Emerson & A. Sistla (1984): *Deciding full branching time logic*. *Inf. and Control* 61, pp. 175 – 201. Available at `http://dx.doi.org/10.1016/S0019-9958(84)80047-9`.

[6] E.A. Emerson (1990): *Temporal and modal logic*. In J. van Leeuwen, ed.: *Hbk of Th. Comp. Sci.*, B, Elsevier.

[7] O. Friedmann, M. Latte & M. Lange (2010): *A Decision Procedure for CTL* Based on Tableaux and Automata*. In: *IJCAR'10*, pp. 331–345. Available at `http://dx.doi.org/10.1007/978-3-642-14203-1_28`.

[8] Valentin Goranko & Dmitry Shkatov (2009): *Tableau-Based Procedure for Deciding Satisfiability in the Full Coalitional Multiagent Epistemic Logic*. In Artëmov & Nerode [1], pp. 197–213. Available at `http://dx.doi.org/10.1007/978-3-540-92687-0_14`.

[9] Orna Kupferman & Moshe Y. Vardi (2005): *Safraless Decision Procedures*. In: *FOCS*, IEEE Computer Society, pp. 531–542. Available at `http://doi.ieeecomputersociety.org/10.1109/SFCS.2005.66`.

[10] M. Lange & C. Stirling (2000): *Model Checking Games for CTL**. In: *In ICTL'00*, pp. 115–125.

[11] J. McCabe-Dansted (2011): *A Rooted Tableau for BCTL*. *Electr. Notes Theor. Comput. Sci.* 278, pp. 145–158. Available at `http://dx.doi.org/10.1016/j.entcs.2011.10.012`.

[12] Fabio Mogavero, Aniello Murano, Giuseppe Perelli & Moshe Y. Vardi (2012): *What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic*. In Maciej Koutny & Irek Ulidowski, editors: *CONCUR*, *Lecture Notes in Computer Science* 7454, Springer, pp. 193–208. Available at `http://dx.doi.org/10.1007/978-3-642-32940-1_15`.

[13] A. Pnueli (1977): *The temporal logic of programs*. In: *Proceedings of the Eighteenth Symposium on Foundations of Computer Science*, pp. 46–57. Providence, RI.

[14] M. Reynolds (2001): *An Axiomatization of Full Computation Tree Logic*. *J.S.L.* 66(3), pp. 1011–1057.

[15] M. Reynolds (May, 2013): *A Faster Tableau for CTL*, Long Version*. Technical Report, CSSE, UWA. Available at `http://www.csse.uwa.edu.au/~mark/research/Online/quicktab/`. Implemented reasoner also available here.

[16] Mark Reynolds (2009): *A Tableau for CTL**. In Ana Cavalcanti & Dennis Dams, editors: *FM 2009: Eindhoven, 2009. Proc.*, *Lecture Notes in Computer Science* 5850, Springer, pp. 403–418. Available at `http://dx.doi.org/10.1007/978-3-642-05089-3_26`.

[17] M. Reynolds (2011): *A tableau-based decision procedure for CTL**. *J. Formal Aspects of Comp.*, pp. 1–41. Available at `http://dx.doi.org/10.1007/s00165-011-0193-4`.

[18] M. Vardi & L. Stockmeyer (1985): *Improved Upper and Lower Bounds for Modal Logics of Programs*. In: *17th ACM Symp. on Theory of Computing, Proceedings*, ACM, pp. 240–251. Available at `http://dx.doi.org/10.1145/22145.22173`.