Towards A Tableau for ATL*

Mark Reynolds

Logic and Computation Group,
School of Computer Science and Software Engineering,
The University of Western Australia

Game Theory and Logics Workshop,
University of Western Sydney,
20 April 2012
Outline

1. Introduction
   - Abstract
   - Introduction

2. ATL*
   - Syntax and Semantics
   - Examples of ATL*

3. The Tableau
   - Preliminaries
   - A CTL* Tableau Example
   - Implementation

4. Conclusion
Outline

1. Introduction
   - Abstract
   - Introduction

2. ATL*
   - Syntax and Semantics
   - Examples of ATL*

3. The Tableau
   - Preliminaries
   - A CTL* Tableau Example
   - Implementation

4. Conclusion
Abstract

ATL*, the full alternating-time temporal logic, is a branching-time temporal logic that naturally describes computations of multi-agent system and multiplayer games. It is used to reason about properties such as receptiveness, realizability, and controllability.

We describe ongoing research into providing a sound, complete and usable tableau method for deciding valid formulas in ATL*. It would be the first such tableau.
specifications for systems are like games
logics can provide unambiguous specifications of desired properties or properties of interest
systems can be checked against properties, we can decide if specifications are at all satisfiable, we can build systems to satisfy specifications, we can show that properties follow from specifications
ATL* is one of several logics adequate for providing complicated statements about whether groups of agents (or components) can enforce extended temporal properties
Games vs Specifications

- many desired properties of (open) systems can be posed in terms of a winning condition in a two-player game between the system and the environment
- receptiveness: can every finite safe computation be extended to an infinite live computation regardless of the behaviour of the environment
- realizability: can a reactive system be built that guarantees that its behaviour satisfies a temporal specification regardless of the environment
- controllability: controller to choose transitions to keep an FSM in a pre-determined safe set of states
- module checking: etc
- generalization to a set of components plus environment as the players.
“If there are an infinite number of requests then there will be an infinite number of grants"

Can be formalised in a propositional linear temporal logic as, eg, $G Fr \rightarrow GF g$

Very easy to satisfy in a closed system: just stop making requests after a while.

But that is not what we mean!

We mean that if the environment generates an infinite number of requests then the system must provide an infinite number of grants. (*note)
From a multiprocess distributed system spec" “a processor must eventually be able to read and it must eventually be able to write "

Can be formalised in a propositional linear temporal logic as, eg, $AG(\neg \exists{r} F_{r} \land \neg \exists{w} F_{w})$

Very easy to satisfy in a closed system: make sure that the other processors cooperate.

But that is not what we want.

We mean that processor $a$ must eventually be able to read (and write) regardless of what the other processors do.
Overview of ATL*

- related to the more usual "alternating-time temporal logic" ATL as CTL* is to CTL;
- a propositional branching time temporal logic similar to CTL*
- so it has temporal connectives: next $X$, until $U$ plus path quantifiers;
- the individual one step transitions are determined by the combined choice of a certain fixed number of players
- path quantifiers reflect the ability of coalitions of players to ensure temporal properties along paths via their choices
- introduced by Alur, Henzinger and Kupferman [AHK02]
- applications: in reasoning about games and specifications for open reactive systems
Finite set $\Sigma = \{1, \ldots, k\}$ of players.

At each state $q$ of the game each player $a$ has a certain number $d_a(q) \geq 1$ of choices of move.

If $d_a(q) = 1$ then $a$ has "no choice": there is just one move that they can make. This allows situations such as players taking turns to move (the other players all have no choice).

The choices for player $a$ in state $q$ are $\{0, 1, \ldots, d_q(a) - 1\}$.

The next state $q' = \delta(q, j_1, \ldots, j_k)$ of the game is determined by the vector of choices $j_i$ of each of the players $i \in \Sigma$ ($\delta$ is the game transition function).
Path Quantifiers

For each subset $A \subseteq \Sigma$ of players, the quantifier $\ll A \gg \alpha$ (applied to formula $\alpha$) is true at a state (i.e. along all paths starting at that state) iff the players in $A$ can together choose moves (while the game is in this state and ever afterwards) to make $\alpha$ hold along the path of play from now onwards regardless of what moves the players in $\Sigma \setminus A$ make at any step.

(Formalised via strategy functions: and strategies may allow unbounded memory).
Formsulas of ATL* are evaluated in...

**Definition**

A *concurrent game structure* is a tuple $M = (k, Q, \Pi, \pi, d, \delta)$ s.t.:

1. $\{1, \ldots, k\}$ is the set of players $k \geq 1$;
2. $Q$ is the non-empty finite set of *states*;
3. $\Pi$ is a set of propositions;
4. $\pi$ is the labeling of states, $\pi(q) \subseteq \Pi$;
5. $d_a(q) \geq 1$ is number of moves available to player $a$ at state $q$;
6. $\delta(q, j_1, \ldots, j_k) \in Q$ is the next state if each player $i = 1, \ldots, k$ chooses move $j_i$ in state $q$.

Formulas are defined along *fullpaths* in $(S, R)$: an infinite sequence $\langle q_0, q_1, q_2, \ldots \rangle$ of states such that for each $i$, $q_{i+1}$ is a possible next state after $q_i$. 
The formulas of ATL* are built from the atomic propositions in $\Pi$ recursively using classical connectives $\neg$ and $\land$ as well as the temporal connectives $X$, $U$ and a path quantifier $\ll A \gg$ for each $A \subseteq \Sigma$: if $\alpha$ and $\beta$ are formulas then so are $X\alpha$, $\alpha U \beta$ and $\ll A \gg \alpha$.

As well as the standard classical abbreviations, $\text{true}$, $\lor$, $\to$, $\leftrightarrow$, we have linear time abbreviations $F\alpha \equiv \text{true} U \alpha$ and $G\alpha \equiv \neg F \neg \alpha$. 
Semantics

Write $M, \sigma \models \alpha$ iff the formula $\alpha$ is true of the fullpath $\sigma$ in the structure $M = (k, Q, \Pi, \pi, d, \delta)$ defined recursively by:

- $M, \sigma \models p$ iff $p \in g(\sigma_0)$, for any $p \in \mathcal{L}$
- $M, \sigma \models \neg \alpha$ iff $M, \sigma \not\models \alpha$
- $M, \sigma \models \alpha \land \beta$ 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 some $i \geq 0$ such that $M, \sigma_{\geq i} \models \beta$ and for each $j$, if $0 \leq j < i$ then $M, \sigma_{\geq j} \models \alpha$
- $M, \sigma \models \ll A \gg \alpha$ iff (roughly) the players in $A$ can choose moves so that regardless of other moves, for all resulting fullpaths $\sigma'$ with $\sigma_0 = \sigma'_0$ we have $M, \sigma' \models \alpha$
\[ \alpha \text{ is valid in } ATL^* \text{ iff for all game structures } M, \text{ for all fullpaths } \sigma \text{ in } M, \text{ we have } M, \sigma \models \alpha. \]

\[ \alpha \text{ is satisfiable iff } \not\models \neg \alpha. \]
**ATL* basics**

- Often write $\ll\{a, \ldots, b\}\gg$ as $\ll a, \ldots, b\gg$.

- Write $\ll\ll\alpha\gg = \ll\{\emptyset\}\gg \alpha$.

- Note: $\ll\ll\alpha\gg$ means every fullpath from here satisfies $\alpha$. (Like $A\alpha$ in CTL*).

- If $\Sigma = \{1, 2, \ldots, k\}$, the set of all players, then $\ll\Sigma\gg \alpha$ means there is some fullpath from here satisfying $\alpha$. (Like $E\alpha$ in CTL*).

- Can show $\ll\Sigma\gg \alpha$ iff $\neg \ll\ll\neg\alpha\gg$.
Outline

1. Introduction
   - Abstract
   - Introduction

2. ATL*
   - Syntax and Semantics
   - Examples of ATL*

3. The Tableau
   - Preliminaries
   - A CTL* Tableau Example
   - Implementation

4. Conclusion
ATL* variations

- ATL is a syntactic restriction (as CTL is of CTL*), i.e. temporal operators $X$, $U$, $F$, $G$ only immediately under $≪A≫$;
- (weaker still) the non-temporal Coalition Logic of [Pau02]
- stronger languages such as various strategy logics eg [WvdHW07] "ATL with explicit strategies".
ATL* Examples (most not in ATL)

\(\ll a \gg (G Fr \rightarrow GFg)\)

\(\ll\bigotimes\ G(\ll a \gg Fr \land \ll a \gg Fw)\)

\(\ll 1 \gg Xp \land \neg \ll 1, 2 \gg Xq \land \ll 3 \gg X \neg q \land \ll 2 \gg Xq\)

\(\ll\bigotimes\ G(p \rightarrow \ll 1 \gg Fp) \rightarrow (p \rightarrow \ll 1 \gg GFp)\)

\(\ll 1 \gg \ll 2 \gg Xp \rightarrow \ll 1, 2 \gg Xp\)
ATL* Reasoning

- Validity decidable [Sch08] (ATL is: in [GvD06]).
- Complexity of checking satisfiability? ATL* is 2EXPTIME-complete [Sch08] (as CTL* [EJ88, VS85], CTL and ATL are EXPTIME-complete [FL79], [GvD06])
- Hilbert-style axiomatization for CTL* in [Rey01], of ATL in [GvD06]. None for ATL*.
- ATL* model-checking is double exponential [AHK02].
- But still there is interest in finding approaches which are more straightforward, or more traditional, or more amenable to human understanding, or yield meaningful intermediate steps, etc ...
Tableaux

- Tableaux popular
- Substantial amount of work on applying them to temporal logics: see [Gor99] and [RD05].
- Can be presented in an intuitive way; often suitable for automated reasoning; often not hard to prove complexity results for their use; often can quickly build models of satisfiable formulas even though the worst case performance is bad;
- First used for modal logics in [HC68] and [Fit83] and there has been much work since on tableaux for temporal logics [Wol85, EC82, EH85, Sch98].
- For CTL* has been a long standing open problem but now appearing in [Rey09], [Rey11] and [FLL10].
Why a Tableau?

- (and despite double exponential lower bound on the complexity)
- experienced tableau practitioners use techniques to practically useful implementations
- basis for searching for more practical sublanguages
- assisting human-guided derivations on bigger tasks.
- basis of proofs of correctness for alternative reasoning techniques like resolution or rewrite systems.
- may assist with model-checking and program synthesis tasks.
- may be extended to predicate reasoning.
Most work on temporal tableaux moves away from traditional tree-shaped tableau standard for temporal logics is to start with a graph and repeatedly prune away nodes, according to certain removal rules, until there is nothing more to remove (success) or some failure condition is detected ([Wol85, EH85]). Return to tree shape for CTL* and ATL*.
The Decision Procedure

- want to decide validity of formula
- start with $\phi$ and determine whether $\phi$ is satisfiable in ATL* or not.
- To decide validity we will simply determine satisfiability of the negation.
- non-deterministically build a tree from root to successors and backtrack on our choices if there is no way to continue building.
- conditions for a successful termination.
- but nodes labelled with sets of sets of formulas
Figure: A Tableau Under Construction
Hues and Colours

- From the closure set for $\phi$, which is just the subformulas and their negations,
- we will define a certain set of subsets of the closure set called the **hues** of $\phi$.
- The **colours** of $\phi$ will be certain sets of hues of $\phi$.
- The nodes in our tableau tree will each be labelled with one colour.
- Whether a given colour can label a successor node will be determined by certain conditions on the formulas in the hues in the label colours on the successor and the parent.
Fix $\phi$

**Definition (closure set)**

The *closure set* for $\phi$ is $\text{cl}\phi = \{\psi, \neg\psi | \psi \leq \phi\}$: its subformulas and their negations.
Figure: Close Up of One Node
Example Node

Figure: Example of One Node
Life-cycle of a Node

Figure: Life Cycle
Temporal and Boolean Decompositions

Formulas in hues are broken down. Iterate.

- $\alpha$ and $\neg\alpha$ in a hue means that the tableau process has made a bad choice. Backtrack!
- $\neg\neg\alpha$ in a hue means put $\alpha$ in too.
- $\alpha \land \beta$ in a hue means $\alpha$ and $\beta$ should also go in.
- $\neg(\alpha \land \beta)$ in a hue means either $\neg\alpha$ or $\neg\beta$ should also go in. Make a choice. (When backtracking make the other choice).
- Leave $X\alpha$ and $\neg X\alpha$: nothing to do at this stage
- $\alpha U\beta$ in a hue means we choose to put $\beta$ in or put $\alpha$ in
- $\neg(\alpha U\beta)$ in a hue means put $\neg\beta$ and choose to put $\alpha$ in or not
The Path Split

Rough description (Based on an idea from Goranko).

Count and enumerate the total number $n$ of positive $\pi_j = \ll A \gg \alpha$ and proper negative $\pi_j = \neg \ll A \gg \alpha$ (A not empty) path quantified formulas in the state.

Each player gets $n$ moves so there are $k^n$ “hues”.

A vector will support $\alpha$ according to $\pi_j = \ll A \gg \alpha$ iff all the players in $A$ choose $j$.
A vector will support $\neg \alpha$ according to $\pi_j = \neg \ll A \gg \alpha$ iff all the players not in $A$ choose $j$.

This determines content of the $n^k$ hues. (Consolidate)
Eg, split $\ll 1 \gg Xq, \neg \ll 1 \gg Gq, \ll 1, 2 \gg Xp$.

Three moves each player. Nine hues.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$Xq$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$Xq, \neg Gq$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>$Xq$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$\neg Gq$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$\neg Gq$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$Xp$</td>
</tr>
</tbody>
</table>
After a hue split step for ATL*, we may need to iterate the temporal, boolean and path split steps.

Will it end?
State formula Rules

Node development requires state formulas to be shared across hues.

- State formulas are atoms and $\ll A \gg \alpha$ and their boolean combinations.
- If an atom or its negation appear in one hue then they must be copied to all other hues.
- Similarly any formula of the form $\ll A \gg \alpha$ or a negation.
- This causes need for iteration of splitting. (Not sure how that will work exactly)
The temporal step

When a node is ready to have children, choose a new child node for each hue. Give each new node one empty hue.

- $X\alpha$ in a hue means $\alpha$ goes in the successor hue.
- $\neg X\alpha$ in a hue means $\neg \alpha$ goes in the successor hue.
- $\alpha U \beta$ but not $\beta$ in a hue means $\alpha U \beta$ goes in the successor.
- $\neg (\alpha U \beta)$ and $\alpha$ in a hue means $\neg (\alpha U \beta)$ goes in the successor.
Example Tableau

Figure: Start of a Tableau for $\theta$
Tableaux can be built node by node.

Work within a hue, the path split, and the temporal one step have been discussed.

Another challenge faced for ATL* as for CTL* is to do with looping and repetition.

In our tableaux there will be “good looping" and “bad looping".
...an ancestor of a node can serve as the successor of that node.

Our tableaux not strictly tree-shaped: allowed to loop up from nodes to their ancestors.

way of making a finite model which has infinite fullpaths.

subtle conditions determine when we are allowed to loop up
... we determine that we have extended a branch in a certain way which exhibits too much repetition with no allowed loops back up, and so no prospect of terminating this development in a successful and finite way.

there will be subtle rules to determine when we have such repetition.

allows us to stop constructing a branch and backtrack to a previous choice, or fail in a finite way.
The tableaux we construct will be roughly tree-shaped albeit the traditional upside down tree with a root at the top: predecessors and ancestors above, successors and descendants below.

However, we will allow up-links from a node to one of its ancestors.

Each node will be labelled with a colour, with the hues ordered and, unless it is a leaf, it will have one (ordered) successor for each hue.
Consider the example
\[
\theta_{-12} = \neg \theta_{12} = \neg (AG(p \rightarrow EXp) \rightarrow (p \rightarrow EGp)).
\]

not satisfiable

The formula has length 30

it has 39 hues and 258 colours.
### CTL* example

<table>
<thead>
<tr>
<th>Hue</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>h28</td>
<td>( { \neg p, \neg Xp, EXp, F\neg p, A\neg Gp, \theta_{12}, AG(p \rightarrow EXp), p \rightarrow EGp, \ldots } )</td>
</tr>
<tr>
<td>h30</td>
<td>( { \neg p, Xp, EXp, F\neg p, A\neg Gp, \theta_{12}, AG(p \rightarrow EXp), p \rightarrow EGp, \ldots } )</td>
</tr>
<tr>
<td>h34</td>
<td>( { p, \neg Xp, EXp, F\neg p, EGp, \neg \theta_{12}, AG(p \rightarrow EXp), \neg(p \rightarrow EGp), \ldots } )</td>
</tr>
<tr>
<td>h35</td>
<td>( { p, Xp, EXp, Gp, EGp, \neg \theta_{12}, AG(p \rightarrow EXp), \neg(p \rightarrow EGp), \ldots } )</td>
</tr>
<tr>
<td>h36</td>
<td>( { p, Xp, EXp, F\neg p, EGp, \neg \theta_{12}, AG(p \rightarrow EXp), \neg(p \rightarrow EGp), \ldots } )</td>
</tr>
<tr>
<td>h37</td>
<td>( { p, \neg Xp, EXp, F\neg p, A\neg Gp, \neg \theta_{12}, AG(p \rightarrow EXp), \neg(p \rightarrow EGp), \ldots } )</td>
</tr>
<tr>
<td>h38</td>
<td>( { p, Xp, EXp, F\neg p, A\neg Gp, \neg \theta_{12}, AG(p \rightarrow EXp), \neg(p \rightarrow EGp), \ldots } )</td>
</tr>
</tbody>
</table>

**Figure:** \( \theta_{-12} \): some contents of some hues
Figure: A Partial Tableau for $\theta_{-12}$
Lemma (Soundness)

*If ϕ has a successful tableau then it is satisfiable.*

Has been proved for CTL* version.

Not yet ATL*.
Completeness sketch

Say \((S, R, g), \sigma^0 \models \phi\)

assume WLOG \(S\) has at most some finite number of elements.

unwind to infinite tree

replace repeated nodes by up-links

chop out useless lengths on left-most branches

other branches then dealt with similarly
The construction sketched above would allow us to conclude:

**Lemma**

*If an ATL* formula is satisfiable then it has a successful tableau.*

Thus any algorithm which systematically searches through all possible \( \phi \)-tableaux for a successful one will thus eventually find one for \( \phi \).

The CTL* version is proved but the search could be better.
Implementation of a CTL* tableau search is relatively straightforward

Java prototype implementation shows that for many interesting, albeit relatively small formulas, results can be obtained quickly despite the double exponential theoretical bounds.

Some preliminary results reported in [Rey11].
Summary

- A sound and complete tableau system for CTL* exists (first such)
- Not hard to implement
- Still needs some polishing
- ATL* just needs additional mechanism to handle one step
Reminder: Uses of Tableau

uses of new tableaux systems for CTL*: 

- extract counter-models and formal proofs from tableaux.
- base for developing, or proving correctness of, other techniques such as resolution or term rewriting.
- give indications of simpler more reasonable sub-languages.
- help manual proofs of validity.
- extend to help with reasoning in the predicate case, for example for software verification.
To do

- much work to do to see if this works
- even the CTL* version is still being finalised
- ATL* (just) needs additional mechanism for coping with coalitions while one step is being made
- (underway) [Rey11] reasonably general and quite usable repetition mechanism is proposed and proved correct. But could do better!
- Worst case performance for any such tableau is necessarily bad (at least double exponential) but there is great potential for vast improvements in running times in general or on certain classes of formulas.
Thank you.

Any questions?


M. Reynolds and C. Dixon.
Theorem-proving for discrete temporal logic.

M. Reynolds.
An axiomatization of full computation tree logic.

Mark Reynolds.
A tableau for CTL*.
Mark Reynolds.
A tableau-based decision procedure for CTL*.

S. Schwendimann.
A new one-pass tableau calculus for PLTL.

Sven Schewe.
*Synthesis of Distributed Systems.*

M. Vardi and L. Stockmeyer.
Improved upper and lower bounds for modal logics of programs.


P. Wolper.
The tableau method for temporal logic: an overview.

Alternating time alternating-temporal logic with explicit strategies.