Model Checking Temporal-Epistemic Logic using Alternating Tree Automata

F. Belardinelli ¹  A. V. Jones ²  A. Lomuscio ³

¹Université d’Evry
²Thales UK
³Imperial College London

September 20, 2013
1 Introduction

2 Preliminaries
   - Interpreted Systems
   - the Temporal-Epistemic Logic CTLK

3 Tree Automata
   - Trees
   - Weak Epistemic Alternating Tree Automata
   - Model Checking CTLK

4 The ETAV Model Checker
   - Implementation
   - Evaluation

5 Conclusions and Future Work
Model checking: widely-used technique to verify that a system $S$ satisfies a specification $P$.

- Given a model $M_S$ for $S$ and a formula $\phi_P$ representing $P$, does $M_S \models \phi_P$?
Model checking: widely-used technique to verify that a system $S$ satisfies a specification $P$.

- Given a model $M_S$ for $S$ and a formula $\phi_P$ representing $P$, does $M_S \models \phi_P$?

MC has been studied in relation with Multi-Agent Systems (MAS), a mainstream framework for autonomous, distributed systems [FHMV95].

- However, the state-space explosion problem is particularly acute for MAS.
Model checking: widely-used technique to verify that a system $S$ satisfies a specification $P$.

- Given a model $M_S$ for $S$ and a formula $\phi_P$ representing $P$, does $M_S \models \phi_P$?

MC has been studied in relation with Multi-Agent Systems (MAS), a mainstream framework for autonomous, distributed systems [FHMV95].

- However, the state-space explosion problem is particularly acute for MAS.

Several techniques have been put forward to alleviate this problem: symbolic [LQR09] and bounded MC [HLvdM10], p.o. reduction [LPQ10].
Background

- **Model checking**: widely-used technique to verify that a system $S$ satisfies a specification $P$.
  - Given a model $M_S$ for $S$ and a formula $\phi_P$ representing $P$, does $M_S \models \phi_P$?
- MC has been studied in relation with *Multi-Agent Systems* (MAS), a mainstream framework for autonomous, distributed systems [FHMV95].
  - However, the state-space explosion problem is particularly acute for MAS.
- Several techniques have been put forward to alleviate this problem: *symbolic* [LQR09] and *bounded MC* [HLvdM10], *p.o. reduction* [LPQ10].
- Orthogonal techniques for temporal-only formalisms focus on automata.
  [KVW00]: model checking CTL via alternating tree automata
Introduction

Background

- **Model checking**: widely-used technique to verify that a system \( S \) satisfies a specification \( P \).
  - Given a model \( M_S \) for \( S \) and a formula \( \phi_P \) representing \( P \), does \( M_S \models \phi_P \)?
- MC has been studied in relation with *Multi-Agent Systems* (MAS), a mainstream framework for autonomous, distributed systems [FHMV95].
  - However, the state-space explosion problem is particularly acute for MAS.
- Several techniques have been put forward to alleviate this problem: *symbolic* [LQR09] and *bounded MC* [HLvdM10], *p.o. reduction* [LPQ10].
- Orthogonal techniques for temporal-only formalisms focus on automata.
  - [KVW00]: model checking CTL via alternating tree automata
- Nonetheless, automata-theoretic techniques for temporal epistemic MAS logics have been investigated only partially.
The Contribution

In this talk:

- we put forward an automata-theoretic approach to model check the branching-time epistemic logic CTLK
The Contribution

In this talk:

1. We put forward an automata-theoretic approach to model check the branching-time epistemic logic CTLK.

2. We present and evaluate ETAV, a tool implementing this model checking procedure.
The Contribution

In this talk:

1. we put forward an automata-theoretic approach to model check the branching-time epistemic logic CTLK
2. we present and evaluate ETAV, a tool implementing this model checking procedure.

Main result:

- in selected cases explicit MC returns negative results fast.
  - No need to explore the whole state space.
Interpreted systems: typical formalism for reasoning about knowledge in MAS [FHMV95].

- each agent $A_i \in Ag$ is in some local state $l_i \in L_i$
- $S \subseteq L_1 \times \ldots \times L_m$ is the set of global states

**Definition (Interpreted System)**

An interpreted system is a tuple $\mathcal{P} = \langle R, s_0, \pi \rangle$ such that

(i) $R$ is a non-empty set of runs $\rho : \mathbb{N} \rightarrow S$
(ii) $s_0 \in S$ is the initial state
(iii) $\pi : S \rightarrow 2^{AP}$ is a truth-assignment for atomic proposition in $AP$

Epistemic indistinguishability:

- for every agent $A_i \in Ag$, $(\rho, n) \sim_i (\rho', n')$ iff $\rho_i(n) = \rho_i'(n')$.

IS are temporal epistemic structures, on which we can interpret CTLK.
Let $Ag = \{A_1, \ldots, A_m\}$ be a set of agents.

**Definition (CTLK)**

$$\phi ::= p | \neg \phi | \phi \rightarrow \phi | AX\phi | A\phi U\phi | E\phi U\phi | K_i\phi$$

- CTLK combines the temporal modalities in CTL with an epistemic operator $K_i$ for each agent $A_i \in Ag$. 
Let $A_g = \{A_1, \ldots, A_m\}$ be a set of agents.

**Definition (CTLK)**

$$\phi ::= p | \neg \phi | \phi \rightarrow \phi | AX\phi | A\phi U\phi | E\phi U\phi | K_i\phi$$

- CTLK combines the temporal modalities in CTL with an epistemic operator $K_i$ for each agent $A_i \in A_g$.
- $\overline{K_i} EF$ recover – agent $i$ can’t rule out that the system will eventually recover...
Let $Ag = \{A_1, \ldots, A_m\}$ be a set of agents.

**Definition (CTLK)**

$$
\phi ::= \ p \mid \neg \phi \mid \phi \rightarrow \phi \mid AX\phi \mid A\phi U\phi \mid E\phi U\phi \mid K_i\phi
$$

- CTLK combines the temporal modalities in CTL with an epistemic operator $K_i$ for each agent $A_i \in Ag$.
- $\overline{K}_i \text{ EF recover}$ – agent $i$ can’t rule out that the system will eventually recover . . .
- $\text{EF } K_i \text{ recover}$ – . . . but only when this happens she will be sure of this fact.
Let $A_{gt}$ be the set $Ag \cup \{t\}$.

**Definition (Tree)**

An $A_{gt}$-tree is a set $T \subseteq (\mathbb{N} \times A_{gt})^*$ s.t. if $x \cdot (c, j) \in T$ and $(c, j) \in \mathbb{N} \times A_{gt}$ then

- $x \in T$
- for all $0 \leq c' < c$, also $x \cdot (c', j) \in T$

A $\Sigma$-labelled tree is a pair $\langle T, V \rangle$ where $T$ is a tree and $V : T \rightarrow \Sigma$.

**Lemma**

*Given an IS $\mathcal{P}$, the $S$-labelled tree $\langle T_\mathcal{P}, V_\mathcal{P} \rangle$ obtained by unwinding $\mathcal{P}$ is s.t.*

$$T_\mathcal{P} \models \phi \iff \mathcal{P} \models \phi$$
Example 1 – Unwinding an Interpreted System

Consider the IS $\mathcal{P}$ with $Ag = \{1, 2\}$. 

The $S$-labelled tree $\langle T_{\mathcal{P}}, V_{\mathcal{P}} \rangle$ unwinding $\mathcal{P}$ can be given as
Weak Epistemic Alternating Tree Automata

Extension of Alternating Tree Automata [MSS86], i.e., non-deterministic tree automata endowed with a weakness partition.

Definition (Alternating Tree Automaton)

An *alternating tree automaton* is a tuple $\mathcal{A} = \langle \Sigma, D, Q, \delta, q_0, Ag_t, F \rangle$ such that

(i) $\Sigma$ and $Ag_t$ are defined as above

(ii) $D \subseteq \mathbb{N}$ is a finite set of *degrees* (i.e., branching factors)

(iii) $Q$ is a set of *states* endowed with a *weakness partition*

(iv) $q_0 \in Q$ is the *initial state*

(v) $F \subseteq Q$ is the set of *accepting states*

(vi) $\delta : Q \times \Sigma \times D^{\lvert Ag_t \rvert} \rightarrow B^+(\mathbb{N} \times Ag_t \times Q)$ is the *transition function*

Acceptance is defined w.r.t. a Büchi acceptance condition.
To model check a CTLK-formula $\phi$ on a IS $\mathcal{P}$:
To model check a CTLK-formula $\phi$ on a IS $P$:

1. we construct a WEAA $A_{D,\psi} = \langle 2^P, D, cl(\psi), \delta, \psi, Ag_t, F \rangle$ that accepts all and only the $D$-trees satisfying $\psi$. 

$\Rightarrow$ Compare the situation with alternating tree automata.
To model check a CTLK-formula $\phi$ on a IS $\mathcal{P}$:

1. we construct a WEAA $\mathcal{A}_{D,\psi} = \langle 2^P, D, cl(\psi), \delta, \psi, Agt, F \rangle$ that accepts all and only the $D$-trees satisfying $\psi$.

2. we build the product automaton $\mathcal{A}_{P,\psi}$ for $\mathcal{A}_{D,\psi}$ and $\langle T_P, V_P \rangle$.

By extending the results in [KVW00] all these steps can be performed in linear time.
Model Checking CTLK

To model check a CTLK-formula $\phi$ on a IS $\mathcal{P}$:

1. we construct a WEAA $A_{D,\psi} = \langle 2^P, D, cl(\psi), \delta, \psi, Agt, F \rangle$ that accepts all and only the $D$-trees satisfying $\psi$.

2. we build the product automaton $A_{\mathcal{P},\psi}$ for $A_{D,\psi}$ and $\langle T_{\mathcal{P}}, V_{\mathcal{P}} \rangle$.

3. the language $L(A_{\mathcal{P},\psi})$ is non-empty iff the tree $\langle T_{\mathcal{P}}, V_{\mathcal{P}} \rangle$ is accepted by $A_{D,\psi}$, i.e., iff $\psi$ is true in $\mathcal{P}$.
To model check a CTLK-formula $\phi$ on a IS $\mathcal{P}$:

1. we construct a WEAA $A_{D,\psi} = \langle 2^P, D, cl(\psi), \delta, \psi, Agt, F \rangle$ that accepts all and only the $D$-trees satisfying $\psi$.

2. we build the product automaton $A_{P,\psi}$ for $A_{D,\psi}$ and $\langle T_P, V_P \rangle$.

3. the language $L(A_{P,\psi})$ is non-empty iff the tree $\langle T_P, V_P \rangle$ is accepted by $A_{D,\psi}$, i.e., iff $\psi$ is true in $\mathcal{P}$.

By extending the results in [KVW00] all these steps can be performed in linear time.

⇒ Compare the situation with alternating tree automata.
Example 2 – from CTLK to WEAA

- Consider the CTLK formula $\varphi' = AGK_1K_2p$.
- Put $\varphi'$ into NNF with all abbreviations expanded: $\varphi = A\left(\text{false}\overline{U}K_1K_2p\right)$.
- The closure of $\varphi$ is $cl(\varphi) = \{\varphi, K_1K_2p, K_2p, p\}$.
- The accepting states are $F = \{\varphi, K_1K_2p, K_2p\}$.
- We define $A_{D,\varphi} = \langle 2^\{p\}, D, cl(\varphi), \delta, \varphi, F \rangle$ where the transition relation $\delta$ is defined as

<table>
<thead>
<tr>
<th>$q$</th>
<th>$\delta (q, p, k)$</th>
<th>$\delta (q, \emptyset, k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi$</td>
<td>$\bigwedge_{c=0}^{k_t-1} (c, t, \varphi) \land \bigwedge_{c=0}^{k_i-1} (c, i, p) \land \bigwedge_{c=0}^{k_i-1} (c, i, K_ip)$</td>
<td>$\bigwedge_{c=0}^{k_i-1} (c, K_1K_2p) \land \bigwedge_{c=0}^{k_i-1} (c, K_2p) \land \bigwedge_{c=0}^{k_2-1} (c, 2, K_2p)$</td>
</tr>
<tr>
<td>$K_1K_2p$</td>
<td>$\bigwedge_{c=0}^{k_1-1} (c, 1, K_2p) \land \bigwedge_{c=0}^{k_1-1} (c, 1, K_1K_2p)$</td>
<td>$\bigwedge_{c=0}^{k_1-1} (c, 1, K_1K_2p) \land \bigwedge_{c=0}^{k_1-1} (c, 1, K_2p) \land \bigwedge_{c=0}^{k_2-1} (c, 2, K_2p)$</td>
</tr>
<tr>
<td>$K_2p$</td>
<td>$\bigwedge_{c=0}^{k_2-1} (c, 2, p) \land \bigwedge_{c=0}^{k_2-1} (c, 2, K_2p)$</td>
<td>$\bigwedge_{c=0}^{k_2-1} (c, 2, K_2p)$</td>
</tr>
<tr>
<td>$p$</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
Implementation

- **ETAV** – Epistemic Tree Automata Verifier: explicit-state model checker (written in C++).
  - Open source release available from http://bitbucket.org/etav/etav/

- **Approach taken**: depth-first construction of product automaton $A_{P,\psi}$, interleaved with the non-emptiness check.

- If we can decide whether the run is accepting without building the full product automata, ETAV will return this result early and save on computation.

- **Optimisations**:
  1. information on the satisfaction of a formula at a node is reused.
  2. the sibling for a node is constructed iff the current node is not sufficient to decide path acceptance.
  3. the transition relation is constructed only when required.
The Gossip Protocol

\( GP_1 \quad EF \left( \bigwedge_{i \in Ag} \text{complete}_i \right) \)

\( GP_2 \quad K_{G_1} EF \left( \bigwedge_{i \in Ag} \text{complete}_i \right) \)

\( GP_3 \quad AG \left( \text{complete}_{G_1} \rightarrow K_{G_1} AF \left( \bigwedge_{i \in Ag} \text{complete}_i \right) \right) \)

| \(|A|\) | Formula | Memory (KiB) | Time (s) | Nodes |
|------|--------|--------------|---------|-------|
| 3    | \( GP_1 \) | 3336         | 0.002   | 35    |
|      | \( GP_2 \) | 3336         | 0.002   | 66    |
|      | \( GP_3 \) | 3336         | 0.002   | 131   |
| 4    | \( GP_1 \) | 3576         | 0.030   | 69    |
|      | \( GP_2 \) | 3576         | 0.031   | 531   |
|      | \( GP_3 \) | 3576         | 0.029   | 46    |
| 5    | \( GP_1 \) | 452444       | 84.646  | 95    |
|      | \( GP_2 \) | 452308       | 84.573  | 41596 |
|      | \( GP_3 \) | 452100       | 84.649  | 232   |

- the high execution times for \(|A| = 5\) arises from parsing the large, explicitly-declared state space.
The Faulty Train Gate Controller

\[ TGC_1 \quad AG \left( train1\text{-}in\text{-}tunnel \rightarrow EF \neg train1\text{-}in\text{-}tunnel \right) \]
\[ TGC_2 \quad AG \left( \neg train1\text{-}in\text{-}tunnel \lor \neg train2\text{-}in\text{-}tunnel \right) \]
\[ TGC_3 \quad AG \left( train1\text{-}in\text{-}tunnel \rightarrow K_{Train1} \neg train2\text{-}in\text{-}tunnel \right) \]
\[ TGC_4 \quad AG \left( K_{Train1} \left( \neg train1\text{-}in\text{-}tunnel \lor \neg train2\text{-}in\text{-}tunnel \right) \right) \]

<table>
<thead>
<tr>
<th>Depth</th>
<th>Formula</th>
<th>Memory (KiB)</th>
<th>Time (s)</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TGC_1</td>
<td>12020</td>
<td>1.383</td>
<td>308</td>
</tr>
<tr>
<td></td>
<td>TGC_2</td>
<td>12024</td>
<td>1.381</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td>TGC_3</td>
<td>12020</td>
<td>1.384</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td>TGC_4</td>
<td>30600</td>
<td>1.973</td>
<td>298284</td>
</tr>
<tr>
<td>6</td>
<td>TGC_1</td>
<td>7932</td>
<td>0.695</td>
<td>1751</td>
</tr>
<tr>
<td></td>
<td>TGC_2</td>
<td>7932</td>
<td>0.697</td>
<td>1118</td>
</tr>
<tr>
<td></td>
<td>TGC_3</td>
<td>7932</td>
<td>0.695</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>TGC_4</td>
<td>12936</td>
<td>0.838</td>
<td>82098</td>
</tr>
<tr>
<td>W</td>
<td>TGC_1</td>
<td>8910</td>
<td>0.638</td>
<td>27822</td>
</tr>
<tr>
<td></td>
<td>TGC_2</td>
<td>8914</td>
<td>0.625</td>
<td>27140</td>
</tr>
<tr>
<td></td>
<td>TGC_3</td>
<td>9037</td>
<td>0.626</td>
<td>29401</td>
</tr>
<tr>
<td></td>
<td>TGC_4</td>
<td>26414</td>
<td>1.106</td>
<td>307169</td>
</tr>
</tbody>
</table>

- unsatisfiable formulas (depth = 1 or 6) lead to smaller state-spaces.
In this talk:

1. we presented a translation of CTLK into (weak epistemic) alternating automata over trees.
In this talk:

1. we presented a translation of CTLK into (weak epistemic) alternating automata over trees.
2. we showed that the language accepted by the product automaton of a CTLK formula $\phi$ and an IS $\mathcal{P}$ is non-empty iff $\mathcal{P} \models \phi$.

In future work we aim at:

- comparing (fairly) the automata-theoretic approach with existing symbolic techniques.
- implementing a real "on-the-fly" model checking procedure.
- exploring the deployment of partial order reduction techniques, which often rely on an automata-theoretic approach.
In this talk:

1. we presented a translation of CTLK into (weak epistemic) alternating automata over trees.
2. we showed that the language accepted by the product automaton of a CTLK formula $\phi$ and an IS $\mathcal{P}$ is non-empty iff $\mathcal{P} \models \phi$.
3. we implemented this procedure in $\textsc{etav}$, an explicit-state model checker for MAS.

In future work we aim at:

- comparing (fairly) the automata-theoretic approach with existing symbolic techniques.
- implementing a real "on-the-fly" model checking procedure.
- exploring the deployment of partial order reduction techniques, which often rely on an automata-theoretic approach.
Conclusions and Future Work

In this talk:

1. we presented a translation of CTLK into (weak epistemic) alternating automata over trees.

2. we showed that the language accepted by the product automaton of a CTLK formula $\phi$ and an IS $\mathcal{P}$ is non-empty iff $\mathcal{P} \models \phi$.

3. we implemented this procedure in ETAV, an explicit-state model checker for MAS.

In future work we aim at:

- comparing (fairly) the automata-theoretic approach with existing symbolic techniques.
- implementing a real “on-the-fly” model checking procedure.
- exploring the deployment of partial order reduction techniques, which often rely on an automata-theoretic approach.
Questions?
Bibliography

*Reasoning about Knowledge.*

X. Huang, C. Luo, and R. van der Meyden.
Improved Bounded Model Checking for a Fair Branching-Time Temporal Epistemic Logic.
*In Proc. of the 6th Workshop on Model Checking and Artificial Intelligence (MoChArt '10),* 2010.

An automata-theoretic approach to branching-time model checking.

A. Lomuscio, W. Penczek, and H. Qu.
Partial order reductions for model checking temporal epistemic logics over interleaved multi-agent systems.

A. Lomuscio, H. Qu, and F. Raimondi.
MCMAS: A Model Checker for the Verification of Multi-Agent Systems.

D. Muller, A. Saoudi, and P. Schupp.
Alternating automata. the weak monadic theory of the tree, and its complexity.