Model Checking Temporal-Epistemic Logic using Alternating Tree Automata

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

[1]Université d'Evry

[2]Thales UK

[3]Imperial College London

September 20, 2013

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 ϕ_P representing P, does $M_S \models \phi_P$?

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 ϕ_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.

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 ϕ_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 ϕ_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

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 ϕ_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:

1. 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 implementating 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 implementating this model checking procedure.

Main result:

- in selected cases explicit MC returns negative results fast.
  - $\Rightarrow$ No need to explore the whole state space.

Models: Interpreted Systems

Interpreted systems: typical formalism for reasoning about knowledge in MAS [FHMV95].

- each agent $A_i \in Ag$ is in some local state $l_i \in L_i$
- $\mathcal{S} \subseteq L_1 \times \ldots \times L_m$ 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* $\rho : \mathbb{N} \rightarrow \mathcal{S}$

(ii) $s_0 \in \mathcal{S}$ is the initial state

(iii) $\pi : \mathcal{S} \rightarrow 2^{AP}$ 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.

Specification Language: the temporal epistemic logic CTLK

Let $Ag = \{A_1, \ldots, A_m\}$ be a set of agents.

Definition (CTLK)

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

- CTLK combines the temporal modalities in CTL with an epistemic operator K_i for each agent $A_i \in Ag$.

Specification Language: the temporal epistemic logic CTLK

Let $Ag = \{A_1, \ldots, A_m\}$ be a set of agents.

Definition (CTLK)

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

- CTLK combines the temporal modalities in CTL with an epistemic operator K_i for each agent $A_i \in Ag$.
- $\overline{K}_i \; EF \; recover$ – agent i can't rule out that the system will eventually recover ...

Specification Language: the temporal epistemic logic CTLK

Let $Ag = \{A_1, \dots, A_m\}$ be a set of agents.

Definition (CTLK)

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

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

Trees

Let Ag_t be the set $Ag \cup \{t\}$.

Definition (Tree)

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

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

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

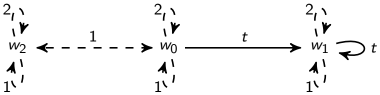
Lemma

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

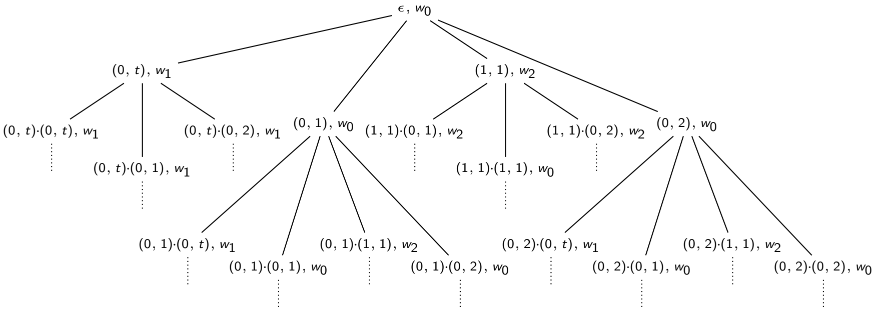
$$T_{\mathcal{P}} \models \phi \quad \textit{iff} \quad \mathcal{P} \models \phi$$

Example 1 – Unwinding an Interpreted System

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



The $\mathcal{S}$-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 $\mathcal{A} = \langle \Sigma, \mathcal{D}, Q, \delta, q_0, Ag_t, F \rangle$ such that

(i) Σ and Ag_t are defined as above

(ii) $\mathcal{D} \subset \mathbb{N}$ 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) $\delta : Q \times \Sigma \times \mathcal{D}^{|Ag_t|} \longrightarrow \mathcal{B}^+(\mathbb{N} \times Ag_t \times Q)$ is the *transition function*

Acceptance is defined w.r.t. a Büchi acceptance condition.

Model Checking CTLK

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

Model Checking CTLK

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

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

Model Checking CTLK

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

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

Model Checking CTLK

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

1. we construct a WEAA $\mathcal{A}_{\mathcal{D},\psi} = \langle 2^P, \mathcal{D}, cl(\psi), \delta, \psi, Ag_t, F \rangle$ that accepts all and only the $\mathcal{D}$-trees satisfying ψ.
2. we build the product automaton $\mathcal{A}_{\mathcal{P},\psi}$ for $\mathcal{A}_{\mathcal{D},\psi}$ and $\langle T_{\mathcal{P}}, V_{\mathcal{P}} \rangle$.
3. the language $\mathcal{L}(\mathcal{A}_{\mathcal{P},\psi})$ is non-empty iff the tree $\langle T_{\mathcal{P}}, V_{\mathcal{P}} \rangle$ is accepted by $\mathcal{A}_{\mathcal{D},\psi}$, i.e., iff ψ is true in $\mathcal{P}$.

Model Checking CTLK

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

1. we construct a WEAA $\mathcal{A}_{\mathcal{D},\psi} = \langle 2^P, \mathcal{D}, cl(\psi), \delta, \psi, Ag_t, F\rangle$ that accepts all and only the $\mathcal{D}$-trees satisfying ψ.
2. we build the product automaton $\mathcal{A}_{\mathcal{P},\psi}$ for $\mathcal{A}_{\mathcal{D},\psi}$ and $\langle T_{\mathcal{P}}, V_{\mathcal{P}}\rangle$.
3. the language $\mathcal{L}(\mathcal{A}_{\mathcal{P},\psi})$ is non-empty iff the tree $\langle T_{\mathcal{P}}, V_{\mathcal{P}}\rangle$ is accepted by $\mathcal{A}_{\mathcal{D},\psi}$, i.e., iff ψ is true in $\mathcal{P}$.

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

$\Rightarrow$ Compare the situation with alternating tree automata.

Example 2 – from CTLK to WEAA

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

q	$\delta(q, p, k)$	$\delta(q, \emptyset, k)$
φ	$\bigwedge_{c=0}^{k_t-1}(c, t, \varphi) \wedge \bigwedge_{c=0}^{k_i-1}(c, i, p) \wedge \bigwedge_{c=0}^{k_i-1}(c, i, K_ip)$	
K_1K_2p	$\bigwedge_{c=0}^{k_1-1}(c, 1, K_2p) \wedge \bigwedge_{c=0}^{k_1-1}(c, 1, K_1K_2p)$	
K_2p	$\bigwedge_{c=0}^{k_2-1}(c, 2, p) \wedge \bigwedge_{c=0}^{k_2-1}(c, 2, K_2p)$	
p	**true**	**false**

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 $\mathcal{A}_{\mathcal{P},\psi}$, 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

$GP_1 \quad EF\left(\bigwedge_{i \in Ag} complete_i\right)$

$GP_2 \quad K_{G1} EF\left(\bigwedge_{i \in Ag} complete_i\right)$

$GP_3 \quad AG\left(complete_{G1} \rightarrow K_{G1} AF\left(\bigwedge_{i \in Ag} complete_i\right)\right)$

$\|A\|$	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 $AG\,(train1\_in\_tunnel \rightarrow EF\neg train1\_in\_tunnel)$
TGC_2 $AG\,(\neg train1\_in\_tunnel \vee \neg train2\_in\_tunnel)$
TGC_3 $AG(train1\_in\_tunnel \rightarrow K_{Train1}\neg train2\_in\_tunnel)$
TGC_4 $AG(K_{Train1}\,(\neg train1\_in\_tunnel \vee \neg train2\_in\_tunnel))$

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

- unsatisfiable formulas (depth $= 1$ or 6) lead to smaller state-spaces.

Conclusions and Future Work

In this talk:

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

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 ϕ and an IS $\mathcal{P}$ is non-empty iff $\mathcal{P} \models \phi$.

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 ϕ 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.

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 ϕ 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

beamericonarticle R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi.
Reasoning about Knowledge.
MIT Press, Cambridge, 1995.

beamericonarticle 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.

beamericonarticle O. Kupferman, M. Y. Vardi, and P. Wolper.
An automata-theoretic approach to branching-time model checking.
Journal of the ACM, 47(2):312–360, 2000.

beamericonarticle A. Lomuscio, W. Penczek, and H. Qu.
Partial order reductions for model checking temporal epistemic logics over interleaved multi-agent systems.
In *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS '10)*, pages 659–666, 2010.

beamericonarticle A. Lomuscio, H. Qu, and F. Raimondi.
MCMAS: A Model Checker for the Verification of Multi-Agent Systems.
In *Proceedings of CAV '09*, volume 5643 of *LNCS*, pages 682–688. Springer, 2009.

beamericonarticle D. Muller, A. Saoudi, and P. Schupp.
Alternating automata. the weak monadic theory of the tree, and its complexity.
In L. Kott, editor, *ICALP*, volume 226 of *Lecture Notes in Computer Science*, pages 275–283. Springer, 1986.