Non-elementary speed up for model checking synchronous perfect recall

Mika Cohen[1] and **Alessio Lomuscio**[1]

Abstract. We consider the complexity of the model checking problem for the logic of knowledge and past time in synchronous systems with perfect recall. Previously established bounds are k-exponential in the size of the system for specifications with k nested knowledge modalities. We show that the upper bound for positive (respectively, negative) specifications is polynomial (respectively, exponential) in the size of the system irrespective of the nesting depth.

1 Past LTLK

We assume that the computational system under analysis is given as a *transition system* $\mathcal{S} = \langle S, R, I_0 \rangle$ consisting of a finite set S of *system states*, a *transition relation* $R \subseteq S \times S$, and a non-empty set $I_0 \subseteq S$ of *initial states*. We assume each state $s \subseteq P$ is a subset of a given set P of atomic propositions; intuitively, state s consists of the atomic facts that hold at s. A *(computation) path* of $\mathcal{S}$ is a finite word $\sigma = s_0, s_1, \ldots, s_t \in S^+$ such that $s_0 \in I_0$ and $(s_i, s_{i+1}) \in R$ for all $0 \leq i < t$.

We assume a finite set Ag of agents; each agent $a \in Ag$ observes a subset $P_a \subseteq P$ of the propositional atoms. Intuitively, the set $s \cap P_a$ represents the local view of agent a at the system state s. We say that two system states s and s' are *equivalent* with respect to agent a, in symbols $s \sim_a s'$, if they agree on the propositions observed by agent a, i.e., if $s \cap P_a = s' \cap P_a$. We say that two paths $\sigma = s_0, s_1, \ldots, s_t \in S^+$ and $\sigma' = s'_0, s'_1, \ldots, s'_t \in S^+$ are *equivalent* with respect to agent a, in symbols $\sigma \sim_a \sigma'$, if they are point-wise equivalent, i.e., $s_0 \sim_a s'_0$, $s_1 \sim_a s'_1$, ..., $s_t \sim_a s'_t$.

Language of past LTLK. LTLK extends LTL with knowledge modalities interpreted through path equivalences (cf. [1]). In this paper we consider the following past-time fragment:

$$\phi \quad ::= \quad p \mid \neg\phi \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid k_a\, \phi \mid \Diamond\phi \mid \ominus \phi \mid \overline{\ominus}\phi$$

where $p \in P$ and $a \in Ag$. The knowledge diamond k_a is read as "agent a considers it possible that", the temporal diamond $\Diamond$ is read as "once", the strong yesterday modality $\ominus$ is read as "yesterday", and the weak yesterday modality $\overline{\ominus}$ is read as "yesterday, if there was a yesterday". We introduce box modalities in the expected way: $K_a\phi$ ("agent a knows that ϕ") abbreviates $\neg k_a \neg\phi$ and $\boxminus\phi$ ("always in the past") abbreviates $\neg\Diamond\neg\phi$. The *knowledge depth* $kd(\phi)$ is the maximal nesting of knowledge modalities: $kd(p) = 0$; $kd(k_a\phi) = 1 + kd(\phi)$; $kd(\Diamond\phi) = kd(\ominus\phi) = kd(\overline{\ominus}\phi) = kd(\neg\phi) = kd(\phi)$; $kd(\phi \wedge \phi') = kd(\phi \vee \phi') = max(kd(\phi), kd(\phi'))$. For example, $kd((k_a k_b)^n p) = kd((K_a K_b)^n p) = 2n$. Given a formula ϕ, the *complement* $\overline{\phi}$ is the formula $\neg\phi$ after double negations have been eliminated and negations have been distributed over conjunctions, disjunctions, and yesterday modalities. The *length* $|\phi|$ of a formula ϕ is the number of symbols in ϕ excluding negations.

We consider two language fragments. We say a formula ϕ is *negative* if any negation in ϕ is only applied to atoms:

$$\phi \quad ::= \quad p \mid \neg p \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid k_a\, \phi \mid \Diamond\phi \mid \ominus \phi \mid \overline{\ominus}\phi$$

A formula ϕ is *positive* if $\overline{\phi}$ is negative, i.e., if ϕ can be obtained from a negative formula by substituting diamonds with boxes:

$$\phi \quad ::= \quad p \mid \neg p \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid K_a\, \phi \mid \boxminus \phi \mid \ominus \phi \mid \overline{\ominus}\phi$$

Synchronous perfect recall semantics. Satisfaction is defined as standard for atomic propositions, boolean operators and temporal modalities, while the knowledge diamonds are interpreted by the corresponding path equivalences as follows:

- $(\mathcal{S}, \sigma) \models k_a\, \phi$ iff $(\mathcal{S}, \sigma') \models \phi$ for some path $\sigma' \sim_a \sigma$

where σ' ranges over paths of $\mathcal{S}$. Informally, $k_a\phi$ holds if ϕ is consistent with the past and present observations of agent a. Given a system $\mathcal{S}$, the extension $[[\phi]]$ of formula ϕ is the set of all computation paths σ of $\mathcal{S}$ such that $(\mathcal{S}, \sigma) \models \phi$. A formula ϕ is valid in the given system, $\mathcal{S} \models \phi$, iff its extension $[[\phi]]$ consists of all computation paths of $\mathcal{S}$.

Example 1.1. *After waiting a random number of days, a sender agent sends a bit value to a receiver agent over a channel that delivers messages immediately or with a delay of one day.*

We model the scenario as a transition system $\mathcal{S}$ with agents a, b and c representing the sender, receiver and channel respectively. We assume an atomic proposition $holds(i, v)$ read as "agent i holds the value v" for each agent $i \in \{a, b, c\}$ and each bit value $v \in \{0, 1\}$. We assume that agent i observes only atomic propositions about agent i itself: $P_i = \{holds(i, 0), holds(i, 1)\}$. We define the transition relation R and the set I_0 if initial states such that the set of possible computation paths of the system is given by the regular expression: $\{holds(a, 0)\}^+ \cdot \{holds(c, 0)\}^? \cdot \{holds(b, 0)\}^* + \{holds(a, 1)\}^+ \cdot \{holds(c, 1)\}^? \cdot \{holds(b, 1)\}^*$.

It can be shown that it is always the case that if the receiver has held the bit value for n days, then $2n$ levels of knowledge have been established, i.e., the positive formula

$$\ominus^n\, holds(b, v) \rightarrow (K_b K_a)^n\, holds(b, v) \tag{1}$$

is valid in $\mathcal{S}$. Moreover, the negative converse of (1) holds: $2n$ levels of knowledge can never be reached in less than n days.

[1] Department of Computing, Imperial College London, UK

2 Non-elementary speed up

The existing model checking algorithms [1] for synchronous perfect recall run in time k-exponential in the size $|S|$ of the system for any past LTLK formula ϕ with knowledge depth k.

Example 2.1. *Existing techniques run in time $2n$-exponential in the size $|S|$ of the system for formula (1) as well as for its converse. In particular, the running time is at least $2^{2^{|S|}}$ for the case when $n = 1$.*

The size $|S|$ of the system is in practice often a large number. Every increase in the knowledge depth k may therefore come at a considerable cost under the existing algorithms. In this section we present improved upper bounds.

2.1 Speed up for positive and negative formulae

The model checking problem for a positive formula can be decided in time polynomial in the size of the system and in time exponential in the length of the formula.

Theorem 2.2. *The model checking problem for a positive formula ϕ can be decided in time $|S|^{2|\phi|}$.*

The model checking problem for a *negative* formula can be solved in time exponential (in a polynomial) in the size of the system and in time doubly-exponential in the length of the formula.

Theorem 2.3. *The model checking problem for a negative formula ϕ can be decided in time $2^{2|S|^{|\phi|+1}}$.*

Example 2.4. *The model checking problem for the positive (1) and its negative converse can be decided in time $|S|^{6(n+1)}$ and in time $2^{2|S|^{3n+4}}$ respectively by Theorems 2.2 and 2.3 (cmp. Example 2.1.)*

We prove Theorems 2.2 and 2.3 by way of the following automata-theoretic characterization of validity

Lemma 2.5. *For any system $\mathcal{S}$ and any negative formula ϕ there exists an automaton $\mathcal{A}_\phi$ of size at most $|S|^{|\phi|}$ with $L(\mathcal{A}_\phi) = [[\phi]]$.*

We construct the automaton $\mathcal{A}_\phi$ for Lemma 2.5 by means of some auxiliary operations on automata. The *scalar product* of an agent $a \in Ag$ and an automaton $\mathcal{A}$ over the alphabet S (the state space of the given system) is the result of replacing ("multiplying") every transition label s in $\mathcal{A}$ with all a-equivalent labels s'.

Definition 2.6 (Scalar multiplication)**.** *Assume an agent $a \in Ag$ and an automaton $\mathcal{A} = \langle S, Q, Q_0, \rho, F\rangle$ over the alphabet S. The product of a and $\mathcal{A}$ is the automaton $a \star \mathcal{A} = \langle S, Q, Q_0, \rho^{a\star}, F\rangle$ over the alphabet S where: $\rho^{a\star}(q, s) = \bigcup\{\rho(q, s') \mid s' \sim_a s\}$.*

In other words, for every transition $q \xrightarrow{s} q'$ in the automaton $\mathcal{A}$ between locations q and q' labelled by state s and for every equivalent state $s' \sim_a s$, there is a transition $q \xrightarrow{s'} q'$ in the automaton $a \star \mathcal{A}$ from q to q' labelled by the equivalent state s'. Consequently, the automaton $a \star \mathcal{A}_\phi$ accepts a word $\sigma \in S^*$ iff the automaton $\mathcal{A}$ accepts some a-equivalent word $\sigma' \sim_a \sigma$.

Remaining auxiliary operations needed are standard. The automaton $\mathcal{A}(F')$ substitutes the set of accepting locations in the automaton $\mathcal{A}$ with the set F'. The automaton $\Diamond\mathcal{A}$ extends each location in $\mathcal{A}$ with a boolean variable that becomes true when we reach an accepting location $q \in F$ and stays true from then on; an extended location is accepting in the automaton $\Diamond\mathcal{A}$ if the boolean variable is true. Finally, the automaton $\mathcal{A}_\mathcal{S}$ corresponding to the given system $\mathcal{S}$ accepts precisely all computation path of $\mathcal{S}$.

Definition 2.7 (Formulae to automata)**.** *The automaton $\mathcal{A}_\phi$ corresponding to a negative formula ϕ is defined inductively as follows:*

- $\mathcal{A}_p := \mathcal{A}_\mathcal{S}([[p]])$.
- $\mathcal{A}_{k_a\phi} := (a \star \mathcal{A}_\phi) \times \mathcal{A}_\mathcal{S}$.
- $\mathcal{A}_{\phi\wedge\phi'} := \mathcal{A}_\phi \times \mathcal{A}_{\phi'}$.
- $\mathcal{A}_{\Diamond\phi} := \Diamond\mathcal{A}_\phi$.

The automata for remaining operators are standard: the automata for $\ominus$ and $\overline{\ominus}$ are defined similarly to the automaton for $\Diamond$ and the automaton for $\vee$ is analogous to the automaton for $\wedge$.

It can be shown that $L(\mathcal{A}_\phi) = [[\phi]]$ and that the size of $\mathcal{A}_\phi$ is at most $|S|^{|\phi|}$. This establishes Lemma 2.5. Theorem 2.2 follows since the model checking problem for a positive specification ϕ can be decided by checking whether $L(\mathcal{A}_{\overline{\phi}}) = \emptyset$ for the negative formula $\overline{\phi}$, given that emptiness is decided in time quadratic in the size (number of locations) of an automaton. In turn, Theorem 2.3 follows from Lemma 2.5 since the model checking problem for a negative specification ϕ can be decided by checking whether $L(((\mathcal{A}_\phi)^d)^c \times \mathcal{A}_\mathcal{S}) = \emptyset$, where $\mathcal{A}^d$ determinizes the automaton $\mathcal{A}$ by means of a subset construction, and $\mathcal{A}^c$ complements the set of accepting locations in $\mathcal{A}$.

2.2 Speed up for arbitrary formulae

Let $ad(\phi)$ be the number of alternations from knowledge diamonds to negations counting from the inside outwards in ϕ. For example, $ad(k_a\neg p) = 0$ and $ad(\neg k_a p) = 1$. In detail, the *alternation depth* $ad(\phi)$ is defined inductively by:

- $ad(k_a\phi) = ad(\phi)$.
- $ad(\neg\phi) = 1 + ad(\phi)$, if ϕ is open.
- $ad(\neg\phi) = ad(\phi)$, if ϕ is closed.

where a formula ϕ is *closed* if every knowledge diamond is within the scope of a negation, and a formula is *open* if not closed.

The model checking problem for arbitrary formulae ϕ can be decided in time $ad(\overline{\phi})$-exponential (in a polynomial) in the size of the system.

Theorem 2.8. *The model checking problem for a formula ϕ can be decided in time $\mathtt{exp}(ad(\overline{\phi}), |S|^{|\phi|+ad(\overline{\phi})})^2$.*

The non-elementary factor in Theorem 2.8, the alternation depth $ad(\overline{\phi})$, may be arbitrarily smaller than the non-elementary factor of existing bounds, the knowledge depth $kd(\phi)$, and it is never worse:

Lemma 2.9. $ad(\overline{\phi}) \leq kd(\phi)$.

To establish Theorem 2.8, we extend the compositional automata construction above with the following construction for negation:

- $\mathcal{A}_{\neg\phi} := (\mathcal{A}_\phi)^c$, if ϕ is closed.
- $\mathcal{A}_{\neg\phi} := ((\mathcal{A}_\phi)^d)^c \times \mathcal{A}_\mathcal{S}$, if ϕ is open.

Thus, we perform a subset construction only when needed, i.e., to restore unambiguity following a scalar multiplication.

Acknowledgements The research described in this paper is partly supported by EPSRC funded project EP/E035655. The authors would like to thank Nir Piterman for valuable comments on earlier drafts of this paper.

REFERENCES

[1] Ron van der Meyden, 'Common knowledge and update in finite environments', *Inf. Comput.*, **140**(2), 115–157, (1998).