Verification of Multi-agent Systems via Combined Model Checking*

Savas Konur Michael Fisher Sven Schewe
Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK
{konur,mfisher,svens}@liverpool.ac.uk

Abstract

Model checking is a well-established technique for the formal verification of concurrent and distributed systems. In recent years, model checking has been extended and adapted for use in multi-agent systems, primarily to enable the formal analysis of BDI systems. While this has been successful, there is a need for more complex logical frameworks in order to verify realistic multi-agent systems. In particular, probabilistic and real-time aspects, as well as knowledge, belief, goals, etc., are required.

However, the development of new model checking tools for complex combinations of logics is both difficult and time consuming. In this paper, we show how model checkers for the constituent logics can be re-used in a modular way when we consider combined logics involving different dimensions. This avoids the re-implementation of model checking procedures. Within this work we define a modular approach, prove its correctness, establish its complexity, and show how it can be used to describe existing combined approaches.

1 Introduction

Model checking is a powerful approach for the formal verification of computer systems. In the area of verification, model checking acquired considerable attention with a stream of results for a variety of temporal logics. Verification of reactive systems by means of model checking techniques is now a well-established area of research [8].

In recent years similar ideas have been applied to the verification of multi-agent systems. Multi-agent systems comprise many different facets, which often makes their formal description hard. Their verification is also difficult, principally because there are often many different dimensions to look at simultaneously, including: autonomous behaviour of agents; beliefs, desires and intentions; teamwork, collaboration and coordination; organizations, norms, societal interactions; uncertainty in sensing and communications; real-time aspects; *etc*. Thus, we may want to represent not just the basic dynamic behaviour of a multi-agent system, but also several of the various aspects mentioned above. Modal logics can specify some of the concepts such as knowledge, beliefs, intentions, norms, and temporal aspects. However, the application of existing model checking tools developed for standard temporal logics, like LTL [32] or CTL [11], to the verification of multi-agent systems [23] is not straightforward. In

*Work supported by EPSRC through project EP/F033567.

order to overcome this problem, researchers have attempted to extend the model checking techniques by adding further operators. These extensions have been carried out based on sophisticated models of autonomous behaviour, uncertainty and interaction [41].

1.1 State of the Art

The model checking of temporal-epistemic aspects of multi-agent systems has received considerable attention. However, the large amount of individual aspects of multi-agent systems has resulted in an unmanagable set of their combinations in the literature.

In [37], for example, the theoretical properties of the model checking problems for epistemic linear temporal logics for interpreted systems with perfect recall are presented. In [35] an approach to model checking for a temporal logic of knowledge, CKL_n (which combines LTL with epistemic logic), was developed. With this approach, *local propositions* provide a means to reduce CKL_n model checking to LTL model checking. In [23] an extension of the method of bounded model checking (one of the main SAT-based techniques) to CTLK (a language comprising both CTL and knowledge operators) is defined, implemented, and evaluated. Work in [18] describes a tool for model checking the logic of knowledge, MCK, which supports a range of knowledge semantics and choices of temporal language. In [33] a tool for model checking epistemic formulae in multi-agent systems is presented. The paper shows how a model checker for temporal models (NuSMV) may be used in the verification of epistemic properties. In [42], approaches to the model checking for the logic CKK_n (which combines CTL and epistemic logic) are provided and algorithms for model checking epistemic operators based on SMV are presented. The work in [28] presents a tool for the automatic verification of temporal and epistemic operators in interpreted systems, MCMAS, supporting ATL operators. A multivalued μK-calculus, an expressive logic for specifying knowledge and time in multi-agent systems, is presented in [24]. In [34], based on the semantics of *interpreted systems with local propositions*, the authors develop an approach to symbolic CKL_n model checking via Ordered Binary Decision Diagrams (OBDDs) and implement the corresponding symbolic model checker, MCTK. A technique for verifying an epistemic logic of branching time, CTLK, based on the model checker NuSMV, is presented in [27]. In [2] multi-agent systems are verified by means of a special action-based temporal logic, ACTLW. Using temporal and epistemic operators, appropriate formulae are created to perform model checking for the system. On the theoretical side [13, 36] study combinations of the epistemic modal logic $\mathsf{S5}_n$ and temporal logics. Finally, Bordini et al. have developed a practical framework for model checking BDI properties of multi-agent programs [6].

Real-time and epistemic aspects of multi-agent systems have also been considered. A real-time temporal knowledge logic, called RTKL, which is a combination of real-time temporal logic and knowledge logic is introduced in [7], and a model checking algorithm for RTKL is presented. Furthermore, cooperation modalities are added to RTKL, obtaining the new logic RATKL that can express not only real-time temporal and epistemic properties but also cooperation properties. Lomuscio, et al. [29] present TECTLK, a logic to specify knowledge and real-time in multi-agent systems, together with an algorithm for bounded model checking based on a discretisation method.

The relations between knowledge and probability were also investigated in the domain of multi-agent systems. In [19] the relations between knowledge and probability were studied, with the work in [9] providing a simplified framework. Delgado et al. proposed an epistemic extension of the probabilistic CTL temporal logic, called KPCTL, allowing epistemic and tem-

poral properties as well as likelihoods of events [10], where, the authors also describe how to extend the PRISM model checker [22] to verify KPCTL formulas over probabilistic multi-agent system models.

1.2 Contribution

The main drawback of the research mentioned above is that numerous combined logics have been introduced to represent different views on multi-agent systems, and this has required the implementation of many different verification systems.

In this paper we present a different approach to model checking multi-agent systems. Instead of introducing new logics for combinations of different aspects, analyzing the model checking problem, and implementing a new checker for a particular combination, we combine logics representing different aspects, using a *generic* model checking method suitable most of these different combinations of logics. In this way, many aspects of multi-agent systems, such as knowledge and time, knowledge and probability, real-time and knowledge, etc., can be represented as a combination of logics, and a combined model checking procedure can be synthesized from model checkers for the component logics. The component logics we have in mind are the logics that refer to the key aspects of multi-agent systems, including:

- logics of time (CTL, LTL, ATL, etc);
- belief/knowledge logics (modal logics KD45, S5, etc);
- logics of goals (modal logics KD, etc);
- probabilistic temporal logics (PCTL, etc); and
- real-time temporal logics (TCTL, etc).

While the formal description of multi-agent systems is essentially multi-dimensional, we generally do not have verification tools for all the appropriate combinations. For example, we might have separate verification tools for logics of knowledge, logics of time, real-time temporal logics, or probabilistic temporal logics, yet we have no tool that can verify a description containing *all* these dimensions.

In this paper we study results and methods for the model checking of combined logics. This can be seen as an extension of the work by Franceschet et al. [16] where model checking for combinations of modal and (simple) temporal logics are considered. Their work deals with combinations of logics whose semantics are defined as standard *Kripke Structures* $\langle W, \mathcal{R}, V \rangle$, where W is the underlying set of worlds/states, V is a valuation function describing the propositions true at each world/state, and $\mathcal{R}$ is the accessibility/transition relation between elements of W. The main drawback of the work in [16] is that their framework does not capture more complex logics whose semantics are not of the above form. Specifically, models of probabilistic temporal logics can also be represented in the $\langle W, \mathcal{R}, V \rangle$ form, but here $\mathcal{R}$ is not just a simple relation between pairs of states/worlds, but also includes a probability distribution for the transitions. Similarly, real-time temporal logics have a relation $\mathcal{R}$ that is extended with a set of clock constraints, and the model itself is extended with a finite set of clocks. It may well be that some probabilistic/real-time temporal logics can be transformed into standard propositional temporal logics, allowing for the application of the techniques of Franceschet et al. [16], but it is likely that even in this case the increase in size in structure will be prohibitive.

In this paper, we extend the framework from [16] such that it can be used for more complex combinations of propositional logics. This allows us to combine logics of time, logics of belief, logics of intentions, probabilistic temporal logics, and real-time temporal logics, etc., in order to provide a coherent framework for the formal analysis of multi-agent systems. We provide a generic model checking algorithm which synthesizes a combined model checker from the model checkers of simpler component logics. We show that the method terminates, and is both sound and complete. We also analyze the computational complexity of the resulting method, and show that the complexity of the synthesized model checker is essentially the supremum of the complexities of the component model checkers. This result suggests that modularity is easier to achieve in model checking than in deductive approaches, where combination often leads to exponential (or worse) complexity.

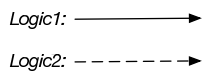
We also show that our combined method is quite useful in determining model checking complexities of some existing logics without the need to study the combined logic. We prove the model checking complexities of the following logics: polynomial bound for the branching time epistemic logic, CTLK [31], PSPACE bound for the linear time epistemic logic, CKL_n [35], polynomial bound for the probabilistic epistemic logic KPCTL [10], and PSPACE bound for a fragment of the real-time epistemic logic TECTLK [29].

Organization of the paper. In Section 2, we provide a brief review of logical combination methods. In Section 3, we present the combined model checking algorithm, and establish its completeness and complexity. In Section 4, we apply our method to some existing logics, and present some complexity results. In Section 5, we provide concluding remarks and discuss potential future research directions.

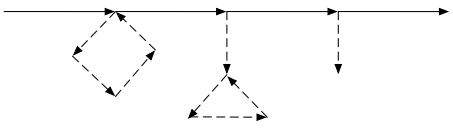
2 Combinations of Logics

Much work has been carried out on the combination of various modal/temporal logics. *Temporalization*, *fusion (or independent combination)*, and *product (or join)* are among the popular forms of logic combination [15, 3, 17, 26].

To provide an overview of different combination forms, let us first imagine we have two logics to combine, *Logic1* and *Logic2*. Further, let us assume that *Logic1* is a temporal logic of some form. Let us represent these graphically as

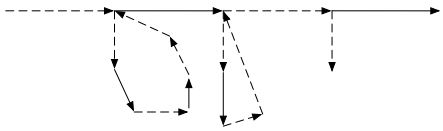


Temporalization. This is a method which adds a temporal dimension to another logic system. In this method an arbitrary logic system is combined with temporal modalities to create a new system. Basically a combination of two logics, A and B, resulting in a logic $\mathrm{A}(\mathrm{B})$ where a pure subformula of B can be treated as an atom within A. Consequently, the combination is not symmetric — the logic A is the main one, but at each world/state described by A we might have a formula of B describing a 'B-world'.



Temporalization is investigated in [14], where the logical properties soundness, completeness and decidability are analyzed. In [16] a model checking procedure is presented for temporalized logics. However, the procedure only covers combining modal propositional temporal logics.

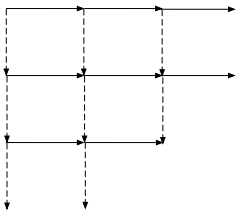
Fusion (or 'Independent Join'). The independent combination of two logic systems put together the expressive power of the two component logic systems in an unrestricted way [14]. In other words, independent combination of two logics over a set of atomic propositions is obtained by the union of the respective sets of connectives and the union of the formation rules of both logics. Unlike temporalization, the fusion of two logics A and B, denoted $\mathrm{A} \oplus \mathrm{B}$, is symmetric. That is, at any state/world we can either take an 'A-step' or a 'B-step'.



Note that the two logics are essentially independent (hence the name). This makes enforcing axioms such as $\vdash\ OP_{\mathrm{A}} OP_{\mathrm{B}} \phi \Leftrightarrow OP_{\mathrm{B}} OP_{\mathrm{A}} \phi$, where OP_{X} is an operator in logic X, impossible. For tighter linking of this kind we use the product combination. The logical properties of fusion, such as soundness, completeness and decidability, are analyzed in [14], while [16] tackles the model checking problem for the fusion of modal temporal logics.

Product (or 'Join'). In both the temporalization and independent join, formulas are evaluated at a single node of a model. On the other hand, in the join of two temporal systems flows of times are considered over a higher dimensional plane. Thus, with the join method it is possible to produce higher-dimensional temporal logics by combining lower dimensional temporal logics.

This product combination is similar to fusion, but with a much tighter integration of the logics. The join of two logics A and B is denoted $\mathrm{A} \otimes \mathrm{B}$, and can be visualized as follows.



Specifically, the operators of the constituent logics tend to be commutative. That is, we add axioms such as $\vdash\ OP_{\mathrm{A}} OP_{\mathrm{B}} \phi \Leftrightarrow OP_{\mathrm{B}} OP_{\mathrm{A}} \phi$. The product construction is investigated in [14], where logical properties of soundness, completeness and decidability are analyzed. In [16] a combined model checking procedure for modal temporal logics is presented.

The decision problem for product logics is typically more complex than that for fused logics, with products of relatively benign logics even becoming undecidable. For this reason, product logics are rarely used in practice, with fusions being the main tool for combining logics in a deductive way. However, we argue that this disadvantage does not extend to model-checking product logics, and that model checking products of logics now becomes a feasible approach.

We end the section by giving some results for the combining methods mentioned above. Most of the work carried out on combining logics has been from a *deducibility* or *expressivity* point of view. To generalize many years of important research, we can say that usually, decidability and axiomatizability properties transfer to both temporalizations and fusions from the constituent logics [5, 21]. Thus, the independent combination of two decidable normal polyadic polymodal logics is decidable [39]. The same result holds for modal logics with the converse operator interpreted over transitive frames [38]. As for one-dimensional temporal logics, it is known that PLTL(PLTL) and PLTL$\oplus$PLTL are decidable [14, 15]. In the case of join we have worse results. For instance, the modal logic S5 is NP-complete, S5$\otimes$S5 is NEXPTIME-complete [30], and S5^3 is undecidable and does not have the finite model property [25]. Moreover, PLTL$\otimes$Km is decidable, but PLTL$\otimes$PLTL is not recursively enumerable [40].

3 Modular Model Checking

In this section we generalize the model checking algorithms for combined logics given in [16] to cover more complex cases, such as knowledge, belief, real-time, and probability, etc. As mentioned above, the semantics of these logics are defined over more general structures than standard Kripke structures. In our extended model checking algorithm, the different model types of combined logics are therefore assumed to have an appropriate hybrid model type.

In the sequel, the following general definition can be used for a propositional logic (we assume a propositional extension of modal logic). The language is built from a countable signature of propositional letters $\mathcal{P} = \{p_1, p_2, ...\}$, the boolean connectives $\wedge$ and $\neg$, a set of operators $OP = \{O_1^{i_1}, .., O_n^{i_n}\}$ with arities $i_1, ..., i_n$ (for $n \in \mathbb{N}$), respectively, and the following formation rules:

1. every propositional letter $p \in \mathcal{P}$ is a formula;
2. if ϕ_1, ϕ_2 are formulas, so are $\neg\phi_1$ and $\phi_1 \wedge \phi_2$;
3. if $O_k^{i_k} \in OP$ and $\phi_1, ..., \phi_{i_k}$ are formulas, so are $O_k^{i_k}(\phi_1, ..., \phi_{i_k})$ (for $k \in \mathbb{N}$).

The other boolean connectives $\vee, \rightarrow, \leftrightarrow$ and the constants $\bot$ and $\top$ can be derived in a standard way. Depending on the type of the logic, e.g., modal, real-time, probabilistic etc., the semantics is defined over different structures, such as *Kripke structures*, *Markov decision processes*, *timed automata*, *interpreted systems* etc. This will be discussed further in Section 4.

It is important to mention that in this section we assume the models with *explicit* sets of states, as in the case of Kripke structures with a discrete labeling function. In Section 4.3, we will discuss the case of models with *non-explicit states*, such as timed automata. For simplicity, we also do not denote the initial states in the structure of a model. This can be added without any problem.

3.1 Temporalization

Let A and B be two propositional logics with $OP(A) \cap OP(B) = \emptyset$. Given that $\langle S_1, T_1\rangle$ is frame for the logic A and $\langle S_2, T_2\rangle$ is frame for the logic B, a model $\mathcal{M}$ for $\mathrm{A}(\mathrm{B})$ is the tuple

$\langle S_1 \cup S_2, T_1, T_2, V \rangle$ with a restriction that $s \in S_1$ is not reachable from $s \in S_2 \backslash S_1$. $V : S_2 \rightarrow 2^{\mathcal{P}}$ is a valuation function assigning S_2 to sets of proposition letters. This restriction ensures that the combination is not symmetric. Another restriction is that the logic A should have a temporal operator in order to temporalize the logic B.

Note that T_1 (resp. T_2) is an n-tuple denoting a relation with an (possibly empty) arbitrary labeling of states and accessibility relations on states. T_1 (resp. T_2) gets a different form according to the type of logic A (resp. B). For example, if A (resp. B) is modal logic, then T_1 (resp. T_2) is defined simply as the accessibility relation R. If, however, A (resp. B) is a probabilistic logic, T_1 (resp. T_2) will be the transition relation with probability labels on edges. In Section 4 we show how T_1 (resp. T_2) can have different forms according to the type of the logic.

For a given finite $\mathrm{A}(\mathrm{B})$-model $\mathcal{M}$ and $\mathrm{A}(\mathrm{B})$-formula ψ, the model checking problem for $\mathrm{A}(\mathrm{B})$ is to check whether there exists $s \in S$ such that $\mathcal{M}, s \models_{\mathrm{A}(\mathrm{B})} \psi$.

Let MC_{A} and MC_{B} be model checkers for the logics A and B, respectively. The algorithm in Figure 1 can be used for model checking $\mathrm{A}(\mathrm{B})$ with the input $\mathcal{M}$ and ψ. The procedure first computes the set of *maximal subformulas* $ms(\psi)$. $ms(\psi)$ is a set of subformulas of ψ with the following condition: Assume $O_1 O_1' \phi$ and $O_1' \phi$ are subformulas of ψ, where O_1, O_1' are operators of a component logic. If $O_1 O_1'(\phi) \in ms(\psi)$, then $O_1'(\phi) \notin ms(\psi)$. The procedure then model checks formulas in $ms(\psi)$ in increasing order wrt. their lengths, and accordingly it extends the valuation V within $\mathcal{M}$. Formulas whose main operator in the language of the logics A and B are resolved by taking advantage of the corresponding model checker.

Figure 1 describes the model checking algorithm for the two logics A and B . Note that, in Figure 1, we consider model checkers as procedures. Namely, a model checker takes a model and a formula ψ as input, and extends the valuation function V within the model to V' such that V' maps states to sets of subformulas of ψ as follows: for every subformula ϕ of ψ and every node s, $V'(s)$ contains ϕ if, and only if, ϕ is true at s in the model. Thus, model checking is here a procedure that extends the valuation within the model to include new truth values for the sub-formulas in question.

3.2 Independent Combination

Let A and B be two propositional logics with $OP(A) \cap OP(B) = \emptyset$. A structure $\mathcal{M}$ for $\mathrm{A} \oplus \mathrm{B}$ is a tuple $\langle S, T_1, T_2, V \rangle$, where T_1 (resp. T_2) is a tuple whose elements depend on the type of A (resp. B). Given that $\mathcal{M}$ is a finite $\mathrm{A} \oplus \mathrm{B}$-structure and ψ is a $\mathrm{A} \oplus \mathrm{B}$-formula, the model checking problem for $\mathrm{A} \oplus \mathrm{B}$ is to check whether there exists $s \in S$ such that $\mathcal{M}, s \models_{\mathrm{A} \oplus \mathrm{B}} \psi$. The algorithm in Figure 1 can again be used for model checking $\mathrm{A} \oplus \mathrm{B}$ with the input $\mathcal{M}$ and ψ.

3.3 Join

Let A and B be two propositional logics with $OP(A) \cap OP(B) = \emptyset$. Given that $\langle S_1, T_1 \rangle$ is a frame for the logic A and $\langle S_2, T_2 \rangle$ is a frame for the logic B, a structure $\mathcal{M}$ for $\mathrm{A} \otimes \mathrm{B}$ is a tuple $\langle S_1 \times S_2, \hat{T}_1, \hat{T}_2, V \rangle$, where $\hat{T}_1$ (resp. $\hat{T}_2$) denotes the relation on $S_1 \times S_2$ (resp. $S_2 \times S_1$), and $V : S_1 \times S_2 \rightarrow 2^{\mathcal{P}}$ is a valuation function.

Given that $\mathcal{M}$ is a finite $\mathrm{A} \otimes \mathrm{B}$-structure and ψ is a $\mathrm{A} \otimes \mathrm{B}$-formula, the model checking problem for $\mathrm{A} \otimes \mathrm{B}$ is to check whether there exists a $s_1 \in S_1$ and a $s_2 \in S_2$ such that

MC$_{\mathrm{AB}}$
Input: $\mathcal{M} = \langle S, T_1, T_2, V\rangle$ and ψ

compute $ms(\psi)$
for every $s \in S$, set $\bar{V}(s) = V(s)$
for every $\phi \in ms(\psi)$ (increasing order of $|\phi|$)
case on the form of ψ
$\phi = p$, $p \in \mathcal{P}$: **skip**

$\phi = \phi_1 \wedge \phi_2$:
for every $s \in S$ **if** $\phi_1 \in V(s)$ **and** $\phi_2 \in V(s)$ **then**
set $V(s) = V(s) \cup \{\phi\}$; $\bar{V}(s) = \bar{V}(s) \cup \{P_\phi\}$

$\phi = \neg\phi_1$:
for every $s \in S$ **if** (**not** $\phi_1 \in V(s)$) **then**
set $V(s) = V(s) \cup \{\phi\}$; $\bar{V}(s) = \bar{V}(s) \cup \{P_\phi\}$

$\phi = \mathbf{O}(\phi_1, ..., \phi_n)$, $\mathbf{O} \in OP_{\mathrm{A}}$:
let $\phi' = \phi$
for every $j \in \{1, ..., n\}$ **replace** $\phi_j \in \phi'$ with P_{ϕ_j}
for every $s \in S$ set $V'(s) = V(s)$
compute $S_1 \subseteq S$ wrt. T_1
MC$_{\mathrm{A}}(\langle S_1, T_1, V'\rangle, \phi')$
for every $s \in S$
if $\phi' \in V'(s)$ **then** set $V(s) = V(s) \cup \{\phi\}$;
$\bar{V}(s) = \bar{V}(s) \cup \{P_\phi\}$

$\phi = \mathbf{O}(\phi_1, ..., \phi_n)$, $\mathbf{O} \in OP_{\mathrm{B}}$:
;; Similar to the case $\phi = \mathbf{O}(\phi_1, ..., \phi_n)$, $\mathbf{O} \in OP_A$

Figure 1: A model checking algorithm MC$_{\mathrm{AB}}$ for the combination of the logics A and B.

$\mathcal{M}, s_1, s_2 \models_{\mathrm{A}\otimes\mathrm{B}} \psi$. The algorithm in Figure 1 can again be used for model checking $\mathrm{A} \otimes \mathrm{B}$ with inputs $\mathcal{M}$ and ψ.

3.4 Correctness and Complexity

Theorem 3.1 (Termination)**.** *Let $\mathcal{M} = \langle S, T_1, T_2, V\rangle$ be a finite structure for the combined logic AB. Assume the model checkers MC$_{\mathrm{A}}$ and MC$_{\mathrm{B}}$ are terminating. Then, the combined model checker MC$_{\mathrm{AB}}$ also terminates.*

Proof. MC$_{\mathrm{AB}}$ computes the set of formulas $ms(\psi)$ in a finite time. The procedure also extends the valuation function V mapping a state to a set of proposition letter to $\bar{V}$ mapping a state to a set of subformulas of ψ in a finite time. Since the model checkers MC$_{\mathrm{A}}$ and MC$_{\mathrm{B}}$ are also terminating, the combined model checker MC$_{\mathrm{AB}}$ terminates. □

Theorem 3.2 (Soundness and Completeness)**.** *Let $\mathcal{M} = \langle S, T_1, T_2, V\rangle$ be a finite model for*

the combination of the logics A *and* B*,* ψ *be a combined formula, and* $\bar{V}$ *be the extended valuation function of* V *(after the termination of MC*$_{AB}$*). Assume the model checkers MC*$_{\mathrm{A}}$ *and MC*$_{\mathrm{B}}$ *are sound and complete. Then, there exists* $s \in S$ *such that* $\psi \in V(s)$ *if, and only if,* $\mathcal{M}, s \models_{AB} \psi$*.*

PROOF. In order to prove the theorem it will be sufficient to show for every subformula ϕ of ψ and every state $s \in S$, $\bar{V}$ contains ϕ if, and only if, ϕ is true at s.

(Soundness) For every $\phi \in ms(\psi)$ and $s \in S$, $\phi \in \bar{V}$ implies $\mathcal{M}, s \models_{\mathrm{AB}} \phi$. We prove this by structural induction on ϕ:

- $\phi = p$ $(p \in \mathcal{P})$: Trivial.
- $\phi = \phi_1 \wedge \phi_2$: In the combined model checking procedure, the formulas in $ms(\psi)$ are model checked in increasing order with respect to their lengths. Therefore, if $\phi \in \bar{V}(s)$, then ϕ_1 and ϕ_2 were already model checked, and it was found that ϕ_1, ϕ_2 are true at s, and therefore $\phi_1 \in \bar{V}(s)$ and $\phi_2 \in \bar{V}(2)$. Therefore, $\mathcal{M}, s \models_{\mathrm{AB}} \phi$.
- $\phi = \neg\phi_1$: Similarly, if $\phi \in \bar{V}(s)$, then we know that $\phi_1 \notin \bar{V}(s)$ and $\phi_1(s)$ is not true. Hence, $\mathcal{M}, s \models_{\mathrm{AB}} \phi$.
- $\phi = \mathbf{O}(\phi_1, ..., \phi_n)$ ($\mathbf{O} \in OP_X$), $X \in \{\mathrm{A}, \mathrm{B}\}$): Let ψ' be a formula obtained by replacing every ϕ_i with P_{ϕ_i} ($1 \leq i \leq n$). That is, $\phi' = \mathbf{O}(P_{\phi_1}, ..., P_{\phi_n})$. If $\phi \in \bar{V}(s)$, then the model checker MC$_X$ found out that ϕ' is true. Since MC$_{\mathrm{AB}}$ model checks the formulas in increasing order with respect to their lengths, we know that $P_{\phi_1}, ..., P_{\phi_n}$ are already true. Hence, $\mathcal{M}, s \models_{\mathrm{AB}} \phi$.

(Completeness) We now show that for every $\phi \in ms(\psi)$ and $s \in S$, $\phi \notin \bar{V}(s)$ implies $\mathcal{M}, s \not\models_{\mathrm{AB}} \phi$. Again, we prove this on the structure of ϕ as follows:

- $\phi = p$ $(p \in \mathcal{P})$: Trivial.
- $\phi = \phi_1 \wedge \phi_2$: Since $\phi \notin \bar{V}(s)$, $\bar{V}$ does not contain either ϕ_1, ϕ_2, or both. This means that ϕ_1 and ϕ_2 were model checked, and at least one of them was found to be false. Therefore, $\mathcal{M}, s \not\models_{\mathrm{AB}} \phi$.
- $\phi = \neg\phi_1$: If $\phi \notin \bar{V}(s)$, then $\phi_1 \in \bar{V}(s)$. This means that ϕ_1 was model checked and it was found to be true. Thus, $\mathcal{M}, s \not\models_{\mathrm{AB}} \phi$.
- $\phi = \mathbf{O}(\phi_1, ..., \phi_n)(\mathbf{O} \in OP_X),\ X \in \{\mathrm{A}, \mathrm{B}\})$: Let $\phi' = \mathbf{O}(P_{\phi_1}, ..., P\phi_n)$. If $\phi \notin \bar{V}(s)$, then the model checker MC$_X$ found that ϕ' is not true. Hence, $\mathcal{M}, s \not\models_{\mathrm{AB}} \phi$. □

We now analyze the computational complexity of the model checker MC$_{\mathrm{AB}}$. The complexity of the combined model checker is the sum of *component model checking cost*, which is the cost of performing component model checkers, and *interaction processing cost*, which is the sum of the cost of processing inputs and outputs of the component model checkers and the cost of operations on the extended valuation function.

Theorem 3.3 (Complexity)**.** *Let* $\mathcal{M} = \langle S, T_1, T_2, V \rangle$ *be a finite model for combination of the logics* A *and* B*,* ψ *be a combined formula, and* C_{A} *and* C_{B} *be the time complexity of the model checkers MC*$_{\mathrm{A}}$ *and MC*$_{\mathrm{B}}$*, respectively. Then, provided that* C_{A} *and* C_{B} *are at least linear in the size of the specification, we can model check the validity of* ψ *in* $\mathcal{M}$ *in time*

$$\mathcal{O}\big(\max\{C_{\mathrm{A}}(|\mathcal{M}|, |\psi|), C_{\mathrm{B}}(|\mathcal{M}|, |\psi|)\}\big).$$

Proof. We first calculate the time required for model checking, which is obviously bounded by $\sum_{\phi \in ms(\psi)} C_{\mathrm{X}}(|\mathcal{M}|, |\phi|)$, where X is the appropriate model checker for the respective subformula $\phi \in ms(\psi)$. By a simple inductive argument along the structure of the fromula, we can show that the sum of the length $\sum_{\phi \in ms(\psi)} |\phi| < 2|\psi|$ of their related simplified subformulas is smaller than twice the length of ψ itself, and each individual.

For $m = \max\{C_{\mathrm{A}}(|\mathcal{M}|, |\psi|, C_{\mathrm{B}}(|\mathcal{M}|, |\psi|)\}$, this provides us (together with the trivial assumption that the cost of model checking is at least linear in the size of the specification) with the straight forward estimation $\sum_{\phi \in ms(\psi)} C_{\mathrm{X}}(|\mathcal{M}|, |\phi|) < 2m$.

We now calculate the interaction processing cost. The computation time of $ms(\psi)$ is linear in the size of ψ, and is therefore bounded by $\mathcal{O}(|\psi|)$. The cost factor of computing or updating the valuation functions V and $\bar{V}$ depends on the form of ϕ. As can be easily seen, the cost factor of the form $\mathbf{O}(\phi_1, ..., \phi_n)$ is higher than those of the forms p, $\phi_1 \wedge \phi_2$ and $\neg\phi_1$. According to the model checking algorithm V is updated $\sum_{\phi \in ms(\psi)} \sum_{s \in S}$ times. The outer sum operator is bounded by $\mathcal{O}(\phi)$, and the inner two sum operators are bounded by $|S|$. Hence, the processing time of V is bounded by $\mathcal{O}(|\psi|.|S|)$. $\bar{V}$ is updated $\sum_{s \in S}$ + $\sum_{\phi \in ms(\psi)} \sum_{s \in S}$ times, which is bounded by $\mathcal{O}(|\psi|.|S|)$. In addition, the replacement of subformulas in ϕ' with proposition letters costs $\mathcal{O}(|\phi|)$, which is bounded by $\mathcal{O}(|\psi|)$. Finally, finding the subset S_1 of S wrt. T_1 costs $|S|.|T_1|$, and similarly finding the subset S_2 of S wrt. T_2 costs $|S|.|T_2|$. Thus, the interaction processing cost is dominated by the component model checking cost. Thus, the total cost of the combined is bounded by $\mathcal{O}\big(\max\{C_{\mathrm{A}}(|\mathcal{M}|, |\psi|), C_{\mathrm{B}}(|\mathcal{M}|, |\psi|)\}\big)$. □

Remark. We note that the combined model checking cost of 'join' is in general higher than either 'temporalization' or 'fusion'. First of all, the state space of the combined model is the product of state spaces of the component logics while, in 'temporalization' and 'fusion', the combined state space is the union of two constituent state spaces. Secondly, the model checking of join has an extra *overhead* cost while calculating $\bar{T}_1$ and $\bar{T}_2$ (See Section 3.3). This cost is bounded by $|T_1|.|S_2| + |T_2|.|S_1|$, which does not change the overall complexity; but requires extra computation time.

4 Example Combinations

In this section, we show how the combination method can be applied to represent some combinations of different aspects of multi-agent systems. The examples demonstrate how simple it becomes with our technique to determine the model checking complexity of these logics (and to synthesize a model checker for them).

4.1 Time + Knowledge

Let us consider the combination of time and knowledge aspects. A typical example is the *fusion* of the temporal logic CTL with the modal logic S5. Typically, the modal operator of

S5 formalizes epistemic concepts. Thus, the combination CTL $\oplus$ S5 can be used to reason about knowledge and time.

Assume the modalities 'group knowledge', 'common knowledge' and 'distributed knowledge' are already defined in S5. The language of CTL $\oplus$ S5 is the smallest set of formulas generated by the following grammar:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}[\phi \mathsf{U} \psi] \mid K_i\phi \mid E_\Gamma\phi \mid C_\Gamma\phi \mid D_\Gamma\phi$$

Other operators ($\neg$, $\wedge$, EF, AX, AG, AU, and AF, etc) can be derived in the usual way. In this grammar, $p \in \mathcal{P}$ is an atomic proposition, and the operators EX, EG and EU are the standard CTL operators. Given a set of agents $A = \{1, ..., n\}$, for any agent $i \in A$, the formula $K_i\phi$ is read as "agent i knows ϕ". Given that $\Gamma \subseteq A$ denotes a group of agents, the form formula $E_\Gamma\phi$ is read as "everybody in group Γ knows ϕ"; the formula $C_\Gamma\phi$ is read as "ϕ is common knowledge in gorup Γ"; the formula $D_\Gamma\phi$ is read as "ϕ is distributed knowledge in group Γ".

Assume the combined model $\mathcal{M}$ for the logic CTL $\oplus$ S5 is a tuple $\langle S, R_t, \sim_1, ..., \sim_n, V\rangle$, where $S = S_1 \times ... \times S_n$ is a set of global states, $R_t \subseteq S \times S$ is a temporal relation, $\sim_i \subseteq S \times S$ is an epistemic accessibility relation for i, and $V : S \rightarrow 2^{\mathcal{P}}$. Here we assume that $(s, s') \in R_t$ iff s' is the result of applying a transition function $t : S \times Act \rightarrow S$ (where $Act \subseteq Act_1 \times ... \times Act_n$ is the set of joint actions) to the global state s and a joint action $act \in Act$. As can easily be observed, $\mathcal{M}$ is an interpreted system, $\langle S, R_t\rangle$ is a Kripke frame for CTL, and $\langle S, \sim_1, ..., \sim_n\rangle$ is a Kripke frame for S5.

Given that σ is a *path*, which is an infinite sequence of states, the semantics of CTL $\oplus$ S5 is defined inductively by:

$\mathcal{M}, s \models p$ iff $p \in V(s)$, for $p \in \mathcal{P}$
$\mathcal{M}, s \models \neg\phi$ iff not $\mathcal{M}, s \models \phi$
$\mathcal{M}, s \models \phi_1 \wedge \phi_2$ iff $\mathcal{M}, s \models \phi_1$ and $\mathcal{M}, s \models \phi_2$
$\mathcal{M}, s \models \mathsf{EX}\phi$ iff $\exists\sigma$ s.t. $\sigma[0] = s$ and $\mathcal{M}, \sigma[1] \models \phi$
$\mathcal{M}, s \models \mathsf{EG}\phi$ iff $\exists\sigma$ s.t. $\sigma[0] = s$ and $(\forall i \geq 0)$ $\mathcal{M}, \sigma[i] \models \phi$
$\mathcal{M}, s \models \mathsf{E}[\phi_1 \mathsf{U} \phi_2]$ iff $\exists\sigma$ s.t. $\sigma[0] = s$ and $(\exists k \geq 0)$ s.t. $\mathcal{M}, \sigma[k] \models \phi_2$ and
$(\forall\, 0 \leq j < k)\ \mathcal{M}, \sigma[j] \models \phi_1$
$\mathcal{M}, s \models K_i\phi$ iff $(\forall s' \in S)\ s \sim_i s'$ implies $\mathcal{M}, s' \models \phi$
$\mathcal{M}, s \models E_\Gamma\phi$ iff $(\forall s' \in S)\ s \sim_\Gamma^E s'$ implies $\mathcal{M}, s' \models \phi$
$\mathcal{M}, s \models C_\Gamma\phi$ iff $(\forall s' \in S)\ s \sim_\Gamma^C s'$ implies $\mathcal{M}, s' \models \phi$
$\mathcal{M}, s \models D_\Gamma\phi$ iff $(\forall s' \in S)\ s \sim_\Gamma^D s'$ implies $\mathcal{M}, s' \models \phi$

where $\sim_\Gamma^E = \bigcup_{i\in\Gamma} \sim_i$, $\sim_\Gamma^D = \bigcap_{i\in\Gamma} \sim_i$, and $\sim_\Gamma^C = (\sim_\Gamma^E)^+$ (the transitive closure of $\sim_\Gamma^E$).

According to our combined model checking procedure, the model checking complexity of CTL $\oplus$ S5 transfers the complexity of the component logics CTL and S5. It is trivial to observe that the model checking problem of CTL $\oplus$ S5 can be solved in polynomial time.

This is interesting since, because without introducing a complex method we solved the model checking problem of a combined logic of time and knowledge. Basically, our result provides an upper bound for the model checking problem of the epistemic logic of branching time, CTLK [31]. As can be easily observed CTL $\oplus$ S5 and CTLK are the same logics, because they have the same grammar and same semantics over the interpreted systems. The model checking problem of the logic CTLK was previously analyzed by bounded model checking [31], by unbounded model checking [23], and by building upon an extension of the model checker NuSMV [27]; but no complexity result was given. So, our result is important in this perspective.

Above we applied our method to combine knowledge and branching time aspects. We can also use our approach to combine knowledge and linear time aspects. Namely, we can define the fusion of the logic LTL and S5, which will result in the same language of the logic CKL$_n$ [35]. We can easily show that the complexity of the model checking LTL $\oplus$ S5 is the same as that of LTL, because the complexity of the LTL model checking is higher than the complexity of the S5 model checking. This is the same result with that of [35] where the model checking problem of CKL$_n$ is reduced to LTL model checking by a method using *local propositions*. The advantage of our approach is that we can prove the model checking complexity just using results from the component logics.

4.2 Probability + Knowledge

In the case of merging the probability and knowledge dimensions of multi-agent systems, we can combine probabilistic CTL, PCTL [20], with the modal epistemic logic S5. PCTL $\oplus$ S5 allows for the expression of epistemic properties, temporal properties and likelihood of events. In PCTL $\oplus$ S5 probabilistic uncertainty is expressed using formulas with nested epistemic and probabilistic operators.

Let $A = \{1, ..., n\}$ be set of agents. The syntax of PCTL $\oplus$ S5 is given in terms of state formulas ϕ and path formulas ψ that are evaluated respectively over states and paths.

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathsf{Pr}_{\alpha r}[\psi] \mid K_i\phi \mid E_\Gamma\phi \mid C_\Gamma\phi \mid D_\Gamma\phi$$

$$\psi ::= \phi \mid \phi \mathsf{U}^{\leq k}\phi \mid \phi \mathsf{U}\phi$$

where $\alpha \in \{\leq, <, \geq, >\}$, $r \in [0, 1]$, $k \in \mathbb{N}$, $i \in A$, $\Gamma \subseteq A$, $p \in \mathcal{P}$ is an atomic proposition, Pr$_{\alpha r}$ is the probability operator, K_i, E_Γ, C_Γ and D_Γ are epistemic operators as described in Section 4.1, and U, U$^{\leq k}$ are path formulas.

Assume the combined model $\mathcal{M}$ for the logic PCTL $\oplus$ S5 is a tuple $\langle S, Act, R_t, \sim_1, ..., \sim_n, V\rangle$, where $S = S_1 \times ... \times S_n$ is a set of global states, Act is a set of actions, $R_t \subseteq S \times Act \times [0, 1] \times S$ is the transition relation, $\sim_i \subseteq S \times S$ is an epistemic accessibility relation for i, and $V : S \to 2^{\mathcal{P}}$. Associated with each state s is a set of enabled actions $Act_s \subseteq Act$. For each state $s \in S$, each enabled action $a \in Act_s$, and every state $s' \in S$, we have exactly one transition $(s, a, P_{(s,a,s')}, s') \in R_t$, for some probability $P_{(s,a,s')} \in [0, 1]$, such that $\sum_{s' \in S} P_{(s,a,s')} = 1$. Thus, at each state, each enabled action determines a probability distribution on the next state. There are no other transitions, so no transitions on disabled actions. We assume every state s has some enabled action, so there are no dead ends.

Given that $\mathcal{M}$ is a combined model for PCTL $\oplus$ S5, $\langle S, Act, R_t, V\rangle$ is a Markov Decision Process (MDP) [12], and $\langle S, \sim_1, ..., \sim_n, V\rangle$ is a Kripke structure for $S5$. The semantics of PCTL $\oplus$ S5 is defined as follows:

$\mathcal{M}, s \models \mathsf{Pr}_{\alpha r}[\psi]$ iff $Prob_s^A(\{\sigma \in Path_s^A \mid \mathcal{M}, \sigma \models \psi\})\ \alpha\ r$, for all $A \in Adv_{\mathcal{M}}$, $\alpha \in \{\leq, <, \geq, >\}$, and $r \in [0, 1]$

Assume the set of paths $\{\sigma \in Path_s^A \mid \mathcal{M}, \sigma \models \phi\}$ is measurable for any path formula ψ, state s and adversary A. The semantics of the path formulas is given below:

$\mathcal{M}, \sigma \models \phi$ iff $\sigma[0] \models \phi$
$\mathcal{M}, \sigma \models \phi \mathsf{U}^{\leq k}\psi$ iff $(\exists i \leq k)\ \mathcal{M}, \sigma[i] \models \psi$ and $(\forall j < i)\ \mathcal{M}, \sigma[j] \models \phi$
$\mathcal{M}, \sigma \models \phi \mathsf{U}\psi$ iff $(\exists k \geq 0)\ \mathcal{M}, \sigma \models \phi \mathsf{U}^{\leq k}\psi$

The semantics of the formulas $p, \neg\phi, \phi_1 \wedge \phi_2$ and epistemic formulas $K_i\phi, E_\Gamma\phi, C_\Gamma\phi, D_\Gamma\phi$ are defined as in Section 4.1.

According to our combined model checking procedure, PCTL $\oplus$ S5 transfers complexities from the component logics PCTL and S5. We know that PCTL model checking over an MDP has polynomial complexity [4]. Therefore, we can find a polynomial upper bound for the model checking of the combined logic PCTL $\oplus$ S5. This is interesting because PCTL $\oplus$ S5 subsumes the logic KPTCL, which was shown to have polynomial model checking complexity [10]. Thus, with our method we established the same complexity result without devising a new model checking algorithm.

4.3 Real-time + Knowledge

In the cases "time + knowledge" and "probability + knowledge", we considered models with *explicit* states, and therefore we could consider the model checkers as procedures which extend the valuation function to map states to sets of subformulas of ψ. However, model checkers for real-time logics, in general, receive models with *non-explicit* states. For example, in timed automata such states are called *locations*, which are just abstract representations. The explicit states can be viewed as the tuples of locations and clock valuations. Therefore, we can no longer consider such a model checker labeling the locations with subformulas which are true. This prevents us from using arbitrary nesting of real-time operators in the combined language. But we can solve this problem partially. We know that if the component real-time model checker returns true for a formula, that formula has to be true in the initial location. Using this fact we can combine these two logics with a limitation that arbitrary nesting of real-time operators are not allowed.

However, we can still use any arbitrary nesting of the operators of the second logic. This allows us to express many interesting properties, which can be useful in multi-agent domain. In case of combination of TCTL [1], a real-time extension of CTL, and S5 we cannot express the formulas such as $EF_{[0,10]}K_aEF_{[0,5]}(p)$; but we can still express the formulas, like $EF_{[0,10]}K_aK_b(p)$, $K_aK_bEF_{[0,10]}(p)$, $K_aEF_{[0,10]}K_b(p)$, etc.

We can now consider the combination of real-time and knowledge aspects. We produce a combined logic, which is the fusion of the logic TCTL and S5. Let us define this combination by considering the limitation mentioned above. Assume $A = \{1, ..., n\}$ be set of agents.The syntax of the resulting logic TCTL $\oplus$ S5 is defined by the following grammar:

$$\phi ::= \quad \top \mid p \mid \psi \mid \neg\phi \mid \phi \wedge \phi \mid K_i\phi \mid E_\Gamma\phi \mid C_\Gamma\phi \mid D_\Gamma\phi \mid \mathsf{E}\psi\mathsf{U}_{\alpha k}\psi \mid \mathsf{A}\psi\mathsf{U}_{\alpha k}\psi$$

$$\psi ::= \quad \top \mid p \mid \neg\psi \mid \psi \wedge \psi \mid K_i\psi \mid E_\Gamma\psi \mid C_\Gamma\psi \mid D_\Gamma\psi$$

where $\alpha \in \{\leq, <, \geq, >\}$, $k \in \mathbb{N}$, $i \in A$, $\Gamma \subseteq A$, and $p \in \mathcal{P}$ is an atomic proposition. Standard abbreviations include $\bot, \phi, \phi \vee \phi, \phi \Rightarrow \phi$, etc. as well as $\mathsf{EF}_{\alpha k}\phi$, $\mathsf{AF}_{\alpha k}\phi$, $\mathsf{EG}_{\alpha k}\phi$ and $\mathsf{AG}_{\alpha k}\phi$.

Assume the combined model for the logic TCTL $\oplus$ S5 is a timed automata $\mathcal{TA}$ (with epistemic relations)

$$\langle S, Act, \mathcal{C}, \rightarrow_t, \mathsf{Inv}, \sim_1, ..., \sim_n, V\rangle$$

where $S = S_1 \times ... \times S_n$ is a set of global states, Act is a finite set of actions, $V : S \mapsto 2^{\mathcal{P}}$ is a valuation function, $\sim_i \subseteq S \times S$ is an epistemic accessibility relation for i, $\mathcal{C}$ is a finite set of clocks, and Inv: $S \mapsto G(\mathcal{C})$ a function assigning an *invariant* to any state. The set

$\rightarrow_t \subseteq S \times Act \times G(\mathcal{C}) \times 2^{\mathcal{C}} \times S$ is a finite set of action transitions: for $s \xrightarrow{g,C}_t s'$, g is the *guard* of the transition, and C is the set of clocks to be reset with the transition.

The semantics of TCTL $\oplus$ S5 can be defined on a *real-time interpreted system* [29]. Namely, a real-time interpreted system for $\mathcal{TA}$ is a tuple $\mathcal{M} = \langle Q, \rightarrow, \bar{\sim}_1, ..., \bar{\sim}_n, \bar{V} \rangle$, where $Q \subseteq S \times \mathbb{R}^{|\mathcal{C}|}$, $\rightarrow \subseteq (S \times \mathbb{R}^{|\mathcal{C}|}) \times (Act \cup \mathbb{R}) \times (S \times \mathbb{R}^{|\mathcal{C}|})$ is the transition relation, $\bar{\sim}_i \subseteq Q \times Q$ is an epistemic relation defined by $(s, v)\bar{\sim}_i(s', v')$ iff $s \sim_i s'$ and $v \simeq v'$, for each agent i and for valuations $v, v' \in \mathbb{R}^{|\mathcal{C}|}$, and $\bar{V} :\mapsto 2^{\mathcal{P}}$ is a valuation function such that $\bar{V}(s, v) = V(s)$ for a clock valuation $v \in \mathbb{R}^{|\mathcal{C}|}$.

Let $\mathcal{M}$ be a real-time interpreted system. Assume $Exec(q)$ is the set of all executions from q. Given $\rho \in Exec(q)$, any finite prefix σ leading to a state q' (denoted $q \overset{\sigma}{\mapsto} q'$) has a *duration*, Time$(q \overset{\sigma}{\mapsto} q')$, defined as the sum of all delays along σ. Let $Pref(\rho)$ be the set of all prefixes rho. The satisfaction relation $\models$ is defined as follows:

$\mathcal{M}, q \models \mathsf{E}\phi\mathsf{U}_{\alpha k}\phi$ iff $\exists \rho \in Exec(q)$ with $\rho = \sigma.\rho' \wedge q \overset{\sigma}{\mapsto} q'$
s.t. Time$(q \overset{\sigma}{\mapsto} q')\alpha\ k$, $\mathcal{M}, q' \models \phi \wedge (\forall q'' <_\rho q')\ \mathcal{M}, q'' \models \phi$ $\mathcal{M}, q \models \mathsf{A}\phi\mathsf{U}_{\alpha k}\phi$ iff $\forall \rho \in Exec(q)$, $\exists \sigma \in Pref(\rho)$ s.t. $q \overset{\sigma}{\mapsto} q'$,
Time$(q \overset{\sigma}{\mapsto} q')\alpha\ k$, $\mathcal{M}, q' \models \phi$ and $(\forall q'' <_\rho q')\ \mathcal{M}, q'' \models \phi$

In $\mathsf{E}\phi\mathsf{U}_{\alpha k}\phi$, the classical U is extended by requiring that ϕ be satisfied within a duration verifying the constraint "αk". The semantics of the formulas $p, \neg\phi, \phi_1 \wedge \phi_2$ and epistemic formulas $K_i\phi, E_\Gamma\phi, C_\Gamma\phi, D_\Gamma\phi$ are defined as in Section 4.1.

The model checking problem of the combination of real-time and epistemic aspects of multi-agent systems was studied in [29], where the fusion of an existential fragment of TCTL and the logic S5, called TECTLK, was introduced. One can notice that TCTL $\oplus$ S5 is a restricted version of the logic TECTLK, such that arbitrary nesting of the formulas $\mathsf{E}\psi\mathsf{U}_{\alpha k}\psi$ and $\mathsf{A}\psi\mathsf{U}_{\alpha k}\psi$ are not allowed. In [29], it was shown that the model checking problem is decidable; but no complexity result was provided. Using our combination method we can prove that that the model checking complexity of a restricted subset of TECTLK, namely TCTL $\oplus$ S5, is the maximum model checking complexity of TCTL and S5, which is PSPACE. Although TCTL $\oplus$ S5 is a subset, it can still express many important properties of knowledge and real-time interaction.

5 Concluding Remarks

In this paper, we presented a modular approach to model checking multi-agent systems. Instead of introducing new logics for combinations of different aspects and studying the resulting model checking problem, we combine logics representing different aspects and use a generic model checking method for different combinations of logics. In this way, many aspects of multi-agent systems, such as knowledge and time, knowledge and probability, real-time and knowledge, etc., can be viewed as simple standard combinations of logics and, without introducing a new logic, a generic combined model checking procedure can be synthesized from model checkers of simpler component logics. This avoids the need for complex re-implementation of model checking procedures. We have also shown that our combined method is quite useful — and often more precise than previous results — for determining the model checking complexity of existing logics without the need of devising a new method.

In Section 4 we showed how our combined model checking method can be used to describe some existing combined approaches that have been applied to agent verification. There are numerous others that could be tackled, including alternating temporal logics, logics of intention, logics of belief, etc. We can also describe new, and as yet un-implemented, combinations. For example, we can combine the probabilistic and cooperation aspects, real-time and intention aspects, belief and uncertainty aspects, etc. Due to space limitation we cannot provide examples of such interactions; but with our modular approach we can analyze different merged dimensions of multi-agent systems.

We are currently working on extending our method to cover the full grammar of the TCTL and S5 combination, to apply the method to more complex, and previously unstudied, combinations and to implement the algorithm to provide a tool.

References

[1] R. Alur, C. Courcoubetis, and D. Dill. Model-Checking in Dense Real-time. *Information and Computation*, 104:2–34, 1993.

[2] M. Bagic, A. Babac, and M. Ciglaric. Verifying Epistemic Properties of Multi-agent Systems via Action-Based Temporal Logic. *Proc. Conf. Comp. Intel. for Modelling, Control and Automation*, pages 470–475, 2008.

[3] B. Bennett, C. Dixon, M. Fisher, E. Franconi, I. Horrocks, and M. de Rijke. Combinations of Modal Logics. *AI Review*, 17(1):1–20, 2002.

[4] A. Bianco and L. D. Alfaro. Model Checking of Probabilistic and Nondeterministic Systems. In *Proc. Symp. Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 1026 of *LNCS*, pages 499–513, 1995.

[5] P. Blackburn and M. de Rijke. Why Combine Logics? *Studia Logica*, 59:5–27, 1997.

[6] R. H. Bordini, L. A. Dennis, B. Farwer, and M. Fisher. Automated Verification of Multi-Agent Programs. In *Proc. 23rd IEEE/ACM Int. Conf. Automated Software Engineering (ASE)*, pages 69–78, 2008.

[7] Z. Cao. Model Checking for Real-Time Temporal, Cooperation and Epistemic Properties. In *Proc. Intelligent Information Processing III*, volume 228 of *International Federation for Information Processing*, pages 63–72, 2007.

[8] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[9] N. de Carvalho Ferreira, M. Fisher, and W. van der Hoek. Specifying and Reasoning about Uncertain Agents. *International Journal of Approximate Reasoning*, 49(1):35–51, 2008.

[10] C. Delgado and M. Benevides. Verification of Epistemic Properties in Probabilistic Multi-Agent Systems. In *Proc. Conf. Multiagent System Technologies (MATES)*, volume 5774 of *LNCS*, pages 16–28, 2009.

[11] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier, 1990.

[12] K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4):1–21, 2008.

[13] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[14] M. Finger and D. M. Gabbay. Adding a Temporal Dimension to a Logic System. *Journal of Logic, Language, and Information*, 1:203–234, 1992.

[15] M. Finger and D. M. Gabbay. Combining Temporal Logic Systems. *Notre Dame Journal of Formal Logic*, 37(2):204–232, 1996.

[16] M. Franceschet, A. Montanari, and M. de Rijke. Model Checking for Combined Logics with an Application to Mobile Systems. *Automated Software Engineering*, 11(3):289–321, 2004.

[17] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Number 148 in Studies in Logic and the Foundations of Mathematics. Elsevier Science, 2003.

[18] P. Gammie and R. van der Meyden. MCK: Model Checking the Logic of Knowledge. In *Proc. Int. Conf. Computer Aided Verification (CAV)*, volume 3114 of *LNCS*, pages 256–259, 2004.

[19] J. Y. Halpern. *Reasoning about Uncertainty*. MIT Press, 2003.

[20] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing*, 6:102–111, 1994.

[21] E. Hemaspaandra. Complexity Transfer for Modal Logic. In *Proc. Logic in Computer Science (LICS)*, 1994.

[22] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *Proc. TACAS'06*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

[23] M. Kacprzak, A. Lomuscio, and W. Penczek. Verification of Multiagent Systems via Unbounded Model Checking. In *Proc. 3rd Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, pages 638–645. IEEE Computer Society, 2004.

[24] B. Konikowska and W. Penczek. Model Checking for Multivalued Logic of Knowledge and Time. In *Proc. 5th Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, pages 169–176. ACM, 2006.

[25] A. Kurucz. S5 x S5 x S5 lacks the finite model property. In *Advances in Modal Logic*, pages 321–327, 2000.

[26] A. Kurucz. Combining Modal Logics. In J. van Benthem, P. Blackburn, and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 869–924. Elsevier, 2007.

[27] A. Lomuscio, C. Pecheur, and F. Raimondi. Automatic Verification of Knowledge and Time with NuSMV. In *Proc. 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1384–1389, 2007.

[28] A. Lomuscio and F. Raimondi. Model Checking Knowledge, Strategies, and Games in Multi-agent Systems. In *Proc. 5th Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, pages 161–168. ACM, 2006.

[29] A. Lomuscio and W. P. B. Woźna. Bounded Model Checking for Knowledge and Real-time. *Artificial Intelligence*, 171(16-17):1011–1038, 2007.

[30] M. Marx. Complexity of Products of Modal Logics. *Journal of Logic and Computation*, 9:221–238, 1999.

[31] W. Penczek and A. Lomuscio. Verifying Epistemic Properties of Multi-agent Systems via Bounded Model Checking. *Fundamenta Informaticae*, 55(2):167–185, 2002.

[32] A. Pnueli. The Temporal Logic of Programs. In *Proc. 18th Symp. Foundations of Computer Science (FOCS)*, pages 46–57. IEEE Computer Society, 1977.

[33] F. Raimondi and A. Lomuscio. A Tool for Specification and Verification of Epistemic Properties in Interpreted Systems. *Electronic Notes in Theoretical Computer Science*, 85(2):179–191, 2004.

[34] K. Su, A. Sattar, and X. Lu. Model Checking Temporal Logics of Knowledge via OBDDs. *Computer Journal*, 50(4):403–420, 2007.

[35] W. van der Hoek and M. Wooldridge. Model Checking Knowledge and Time. In *Proc. 9th Int. SPIN Workshop on Model Checking of Software*, pages 95–111. Springer-Verlag, 2002.

[36] W. van der Hoek and M. Wooldridge. Cooperation, Knowledge, and Time: Alternating-time Temporal Epistemic Logic and its Applications. *Studia Logica*, 75(1):125–157, 2003.

[37] R. van der Meyden and N. V. Shilov. Model Checking Knowledge and Time in Systems with Perfect Recall. In *Proc. Symp. Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 1738 of *LNCS*, pages 432–445, 1999.

[38] F. Wolter. Completeness and Decidability of Tense Logics Closely Related to Logics above K4. *Journal of Symbolic Logic*, 62(1):131–158, 1997.

[39] F. Wolter. Fusions of Modal Logics revisited. In *Proc. Advances in Modal Logic*, CSLI Lecture Notes, pages 361 – 379, 1998.

[40] F. Wolter. The Product of Converse PDL and Polymodal K. *Journal of Logic and Computation*, 10(2):223–251, 2000.

[41] M. Wooldridge. *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., 2001.

[42] L. Wu, K. Su, and Q. Chen. Model Checking Temporal Logics of Knowledge and Its Application in Security Verification. In *Computational Intelligence and Security*, volume 3801 of *LNCS*, pages 349–354, 2005.