




Model Checking Strategic Abilities in Information-sharing Systems

FRANCESCO BELARDINELLI , Imperial College London, United Kindom

IOANA BOUREANU , Surrey Centre for Cyber Security, University of Surrey, United Kindom

CATALIN DIMA , Université Paris-Est Créteil, France





VADIM MALVONE , Télécom Paris, Institut Polytechnique de Paris, France

We introduce a subclass of concurrent game structures (CGS) with imperfect information in which agents are endowed with private data-sharing capabilities. Importantly, our CGSs are such that it is still decidable to model-check these CGSs against a relevant fragment of ATL. These systems can be thought as a generalisation of architectures allowing information forks, that is, cases where strategic abilities lead to certain agents outside a coalition privately sharing information with selected agents inside that coalition. Moreover, in our case, in the initial states of the system, we allow information forks from agents outside a given set A to agents inside this group A . For this reason, together with the fact that the communication in our models underpins a specialised form of broadcast, we call our formalism *A-cast systems*. To underline, the fragment of ATL for which we show the model-checking problem to be decidable over *A-cast* is a large and significant one; it expresses coalitions over agents in any subset of the set A . Indeed, as we show, our systems and this ATL fragments can encode security problems that are notoriously hard to express faithfully: terrorist-fraud attacks in identity schemes.

CCS Concepts: • **Theory of computation** → **Logic and verification**; **Verification by model checking**.

Additional Key Words and Phrases: Logic and Reasoning; Alternating-time Temporal Logic; Formal Specification and Verification; Reasoning about Security Protocols.

ACM Reference Format:

Francesco Belardinelli , Ioana Boureanu , Catalin Dima , and Vadim Malvone . 2018. Model Checking Strategic Abilities in Information-sharing Systems. In *Proceedings of (TOCL)*. ACM, New York, NY, USA, 45 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

The Increasing Importance of Private-Information Sharing & Collusion-encoding.

Sharing information over private channels is a key issue in many AI-inspired ICT applications, from robotics to information systems [55]. Safeguarding privacy is also of utmost concern, also in the context of EU's newly-enforced General Data Protection Regulation (GDPR) [1]. So, being able to explicitly model data-sharing over private channels in multi-agent systems (MAS) or concurrent systems is crucial.

There are numerous ICT applications, such as identity-schemes and distributed ledgers, where the threat of adversaries colluding *privately* with inside accomplices is greater than that of classical, outsider-type attackers. That said, verifying security against collusions-based attacks such as terrorist-frauds [15] in identification schemes is a notoriously difficult problem in security verification [22]. And, even the most recent formalisms attempting this [27] fall short of encapsulating the strategic nature of this property or of the threat-model it views. Instead, [27] looks at correspondence of events

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 ACM.

Manuscript submitted to ACM

in applied π -calculus (i.e., if agents did a series of actions with one result, then later another fact must be the case), where they use arguably crude approximations (based on causalities/correspondence of events in process calculi [20]) of what a collusive attack would be. But, indeed, such attacks resting on not one but two interdependent strategies are notoriously hard to model [22]. Meanwhile, expressive logics for strategic reasoning have been shown effective in capturing intricate security requirements, such as coercion-resistance in e-voting [6, 41, 42, 56]. Thus, the further study of strategic abilities under collusion is of timely interest.

Looking into this, we observe that the typical formalisms for encoding multi-agent systems (MAS), such as interpreted systems [30], support the analysis of strategy-oriented properties. Yet, these systems are generally inspired by concurrent action models and, so, explicit expression therein of data-sharing between agents is difficult to model naturally. Other frameworks, such as dynamic epistemic logic (DEL) [47, 60] or propositional assignments [59], provide agents with more flexibility on data sharing, but do not allow for the full power of strategic reasoning. To fill this gap, the formalism of vCGS was recently proposed in [5]. vCGS supports the explicit expression of private-data sharing in MAS. That is, a vCGS encodes syntactically and in a natural semantics a “MAS with 1-to-1 private-channels”: agent a and agent b have an explicit syntactic/semantic endowment to “see” some of each others’ variables, without other agents partaking in this. However, unfortunately, verifying strategy-based logic specification on vCGS has been shown undecidable under the assumptions of perfect recall and imperfect information [29]. So, it would be interesting and timely, to see if we can maintain the needed feature of private-information sharing that vCGS have and recover the decidability of model checking logics that express strategic abilities.

To sum up, in this paper, *we look at tractable verification of AI-inspired logics of strategic abilities in systems capable of expressing private data-sharing between processes; we show an important decidability result in this space. We do this with the view of applying this to ICT systems where the analysis of strategic behaviour is crucial: e.g., collusion-based attacks in security settings.*

1.1 State of the Art

Alternating-Time Temporal Logic (ATL) [4] is a powerful extension of branching-time logics which indeed provides a framework for reasoning about strategic behaviours in concurrent and multi-agent systems. From the very beginning, ATL was proposed also for agents with imperfect information. In ATL semantics, agents can have memoryless strategies or recall-based/memoryful strategies; in the latter case, they “remember” all their history of moves, whereas in the former – they recall none. Also, ATL with imperfect information comes with several semantic flavours [43] (subjective, objective or common-knowledge interpretation), created by different views on the intricate relationships between its temporal and epistemic aspects.

ATL Decidability. There have been several proposals for decidable fragments of ATL with imperfect information, which arose via some alternative avenues: (a) by imposing a memoryless semantics for the ATL operators [46], approach implemented in the MCMAS tool; (b) by making structural restrictions on the indistinguishability relations inside the game structures, so that one looks at ATL just with distributed knowledge [28], or ATL with hierarchical knowledge [16] or ATL over broadcasting systems [12, 13]; (c) by studying approximations to perfect information [9], to bounded memory [11], or to hybrid techniques [31, 32]. Some other decidable fragments can be further obtained by adapting decidable cases of the distributed-synthesis problem, or the the existence of winning strategies in multi-agent games with imperfect information. We therefore may generalise the decidability of the distributed-synthesis problem in architectures without information forks [33], or utilise the decidability of the problem of the existence of a winning

strategy in multi-player games with finite knowledge gaps [19]. An in-depth discussion of these aspects is provided in Sec. 6.

1.2 Contribution

I. Our Private Data-Sharing Systems. In this paper, we propose a new class of game structures called *A-cast game structures*, where A is a set of designated processes/agents. In *A-cast* game structures, when some outsider (agent not in A) sends some information to some agent in A , the same information is seen by all agents in coalition A .

A-cast game structures are introduced using the formalism of visibility CGS (vCGS) recently proposed¹ in [5]. We can introduce in principle *A-cast* systems directly as game structures, yet we chose to use vCGS simply as they allow for an elegant formalisation of the information-flow between outsiders to the coalition A . So, as a consequence of this presentation choice, *A-cast* game structures can be seen as a specialisation of the vCGS in [5].

A-cast game structures are in fact a proper generalisation of both architectures without information forks [33] and broadcast systems [50]. On the one hand, the extension of [33] comes from the fact that, in *A-cast* game structures, the set of initial states is arbitrary, as well as the epistemic relations defining the information available to agents in initial states. On the other, modelling these features using the setting of distributed architectures in [33], where a single initial state is imposed, requires the Environment agent to send the initial information to each agent by some private channel, hence creating an information fork.

II. Our Decidable Fragment of ATL. We identify a significant class of formulas for which the model-checking problem is decidable on our *A-cast* game structures, as well as some details linked to this. This fragment of *ATL** is composed of formulas which utilise only coalition operators involving a set $B \subseteq A$ of agents, where A is the set of agents/processes describing the *A-cast* system.

To obtain this result, the key point is that two action-histories starting in the same initial state, both being generated by the same joint strategy for coalition A and being in the same common-knowledge indistinguishability class for A are in fact in the same distributed-knowledge indistinguishability class for A . This property allows for the design of a finitary information-set construction for the appropriate multi-player game with imperfect information [18], needed to decide each coalition operator.

III. Our Case Study and Its Need of Verification against Formulae with Nested-ATL Modalities. We provide a case study which shows the relevance of our new decidable class of ATL model-checking. This case study lives in the cyber-security domain: the identity-schemes and threats of collusion-based attacks therein (i.e., terrorist-fraud attack). Concretely, we model the distance-bounding (DB) protocol by Hancke and Kuhn [40] as an *A-cast* vCGS, and the existence of a terrorist-fraud attack on this protocol as an *ATL* formula involving a non-singleton coalition operator. In fact, we also note that the *ATL* formula which specifies the existence of a terrorist-fraud attack in our case study is the first type of formula requiring nesting of *ATL* operators.

Hence, the model-checking algorithm proposed in this paper can be applied to this case study, while other existing decidable model-checking or distributed-synthesis frameworks cannot treat it (due to the formula it requires). The only exception would be the utilisation of a memoryless semantics of *ATL* [46], which would be applicable since our case study is in fact a model in which there exists a finite number of lasso-type infinite runs. Yet, specifying our case-study in a formalism like that of [46] would require an explicit encoding/“massaging” of the agent memory into the model. In

¹vCGS can be seen as a generalisation of Reactive Modules Games with imperfect information [39], in that each agent may dynamically modify the visibility of the atoms she “owns”, by either disclosing or hiding the value of that atom to some other agent.

other words, our algorithm synthesises memoryful strategies and hence allows working with a given model without explicit encoding of agent memory into the model.

1.3 Related Work

The work here presented is related to contributions in distributed and multi-agent systems, including propositional control, reactive modules, broadcast, dynamic epistemic logic. However, it differs from the state of the art in numerous, subtle aspects.

The **coalition logic of propositional control** (CL-PC) was introduced in [59] as a logic to reason about the capabilities of agents in multi-agent systems. It has since been investigated in relation with the transfer of control [58], the dynamic logic of propositional assignments [36], non-exclusive control [34]. Furthermore, in [10] it is proved that shared control can be simulated by means of exclusive control. However, the present contribution differs from most of the works in this line in two key aspects. Firstly, the references above deal mainly with *coalition logic*, which is the “next” $\langle\langle A \rangle\rangle X$ fragment of *ATL*. Secondly, they all assume perfect information on the agents in the system, while here we analyse the arguably more challenging case of imperfect information.

Visibility atoms have been used to model explicitly individual knowledge [35], even though only in [5] they were used in the specific version considered in this paper. In particular, in [35] the authors introduce *influence games* played by agents who try to influence the opinion other agents have on a given set of issues. An atom $vis(i, p)$ expresses the fact that “agent i uses her influence power over issue p ”. A notion of (propositional) visibility is analysed in [47] and the works cited therein. In particular, similarly to what we do here, the indistinguishability relation between epistemic states is defined in terms of the truth values of observed atoms. Still, a key difference with [47] and related works is that they introduce modal operators for visibility as well. As a result, their language is richer as it can express introspection for instance. On the other hand, logics for strategies are not considered in this line of research, while they are the focus of this paper.

Guarded commands have appeared in the framework of **Reactive Modules** [3], which vCGS are reminiscent of. Model checking *ATL* on reactive modules has been investigated in [57], but in a perfect information setting only. More recently, imperfect information in reactive modules has been explored in [37, 38], however mainly in relation with the verification of game-theoretic equilibria, while here we study the verification of full *ATL* under imperfect information.

The semantics that we use also compares to the event models of **Dynamic Epistemic Logic** [60]. The underlying philosophy of event models is to provide a flexible formalism for representing one-step system evolutions in aspects regarding both the way the system’s state and information available to each agent change. However, event models do not focus on the local implementation of actions in the way iCGS do, and therefore they only accommodate a limited type of “strategic” reasoning. Also, they allow for a more refined specification of the way actions influence the information update for each agent, by introducing the indistinguishability relations on events. We uphold that the combination of agent-based action specifications (commands) and visibility atoms allows us to reason about both agent strategies and information updates. More formally, assuming that some multi-agent variant of event models can be defined, in the sense of a “parallel composition” of event models for each agent, any instance of a model-checking problem defined by a Kripke structure, an event model and an *ATL** formula can be translated into a CGS model (and hence into a vCGS model too), in which the epistemic formulas in the preconditions constraining event application could be transferred to the *ATL** formula (which would require then the addition of knowledge operators, or a subjective interpretation of the coalition operators).

In this paper, we will encode the notion of “terrorist-fraud attacks” in security protocols. It intuitively says: in one protocol-run, there exists a winning strategy of a coalition including a protocol attacker such that no matter what strategy is followed by said attacker in a future protocol-run, the latter is no longer winning. So, clearly, this property requires nesting of *ATL* operators to be encoded. Yet, as far as we know, encoding **security properties** has never been done with nested *ATL* modalities. In fact, **security properties** have seldom been encoded in *ATL* at all. we know of but a few examples: properties pertaining to contract-signing [54] and e-voting [6, 41, 56]. Also, most of these prior works do not operate in a perfect recall semantics, and the coalitions expressed do not necessarily pose on private-information sharing, as do ours in the collusion setting.

Symbolic security verification of distance bounding (DB), as opposed to computational analysis [21], has emerged only in the last two years [26, 27, 48]. As the most recent line of this type [27] shows, TF attacks can be expressed therein only as a simple approximation by imposing sequence of events. That is because in the process-algebraic [20] or rewriting-based approaches [49] used therein, collusions cannot be expressed either the system-modelling level or at the property-encoding level. Nonetheless, they do model security more faithfully than we do in Section 4, but this is primarily due to the fact that herein we present our model in a pen-and-paper version. As such, we believe our model to be the first formalism to faithfully lend itself to the analysis of terrorist frauds in secure system.

Finally, the restriction to **broadcast** as mode of communication to retain decidability of the model checking problem has been explored in [12, 13, 45, 50]. We have compared these approaches to ours in Sec. 6. More generally, other restrictions on iCGS have been investigated recently, notably the hierarchical systems in [16, 17], which are nonetheless orthogonal to the present setting. In particular, notice that no hierarchy is assumed on the observations of agents in vCGS, nor anything similar appears in standard TF attacks.

Previous Work by the Same Authors. Multi-agent systems with some form of private data sharing have been studied by the same authors in [5], where the formal setting is also based on reactive modules. However, content-wise the similarities end there. Indeed, the main contribution of [5] is a formula-based notion of model reduction, which is only provided for *ATL*, a proper fragment of the logic *ATL** discussed here. Moreover, the definition of *A*-cast as mode of communication among agents is original of this work, including the related decidability result for the model checking problem under the assumption of perfect recall and imperfect information. None of these questions is tackled in [5].

Previous work by some of the present authors on the notion of bisimulation used in Sec. 5 has appeared in [6, 8], with some noticeable differences. For instance, bisimulations in [6, 8] are indexed to a unique coalition; whereas here we allow bisimulations between possibly different coalitions. This generalisation is key for the reduction to two-players games in Sec. 5.2. Moreover, [6, 8] focus on truth-preservation results, whereas decidability problems are not investigated.

Structure. In Section 2, we present Reactive Modules with imperfect information (RMI) and its notion with visibility control, recall notions on Alternating-time Temporal Logic (*ATL*) and concurrent game structures with imperfect information (iCGS) [4], and provide a reduction of RMI to iCGS. In Section 5, we introduce turned-based iCGS and define a truth-preserving relation of bisimulation on them. In Section 5, we present the main decidability result of model checking *ATL* on top of *A*-cast systems. In Section 4, we show how use the logic and result here to check the existence of collusion-based attacks in secure systems.

2 STRATEGIC REASONING FOR REACTIVE MODULES

In this section we provide a notion of agent in the framework of the Simple Reactive Modules Language [57] in the context of imperfect information (RMI) (Sec. 2.1). Then, we recall the syntax of Alternating-time Temporal Logic

(ATL) [4] (Sec. 2.2). Furthermore, in Sec. 2.3 we present an embedding of RMIs into Concurrent Game Structures with imperfect information (iCGSs), which is used in Sec. 2.4 to provide an interpretation of ATL over RMI.

We denote the length of a tuple r as $|r|$, and its i th element as either r_i or $r[i]$. For $i \leq |r|$, let $r_{\geq i}$ or $r[\geq i]$ be the suffix $r_i, \dots, r_{|r|}$ of r starting at r_i , and let $r_{\leq i}$ or $r[\leq i]$ be the prefix r_1, \dots, r_i of r ending at r_i . The prefix relation on histories is then defined as usual: $h \leq k$ iff $k[\leq |h|] = h$. Furthermore, we denote with $\text{last}(r)$ the last element $r_{|r|}$ of r . Hereafter, we assume a finite set $Ag = \{1, \dots, m\}$ of *agents* and an infinite set $AP = \{p_1, p_2, \dots\}$ of *atomic propositions* (or atoms).

Given a set Vars of variables with domain $\text{Dom}(\text{Vars})$, an *evaluation* is a function $I : \text{Vars} \mapsto \text{Dom}(\text{Vars})$. Then, for variable $v \in \text{Vars}$ and (truth) value $\text{tv} \in \text{Dom}(\text{Vars})$, we introduce identities $v = \text{tv}$ and define $I \models v = \text{tv}$ iff $I(v) = \text{tv}$. Notice that such identities allow us to represent multi-valued logics in two-valued Boolean logic. For example, the fact that a propositional formula $p \wedge q$ has value undefined uu in Kleene's three-valued logic [44] according to some evaluation I (that is, $(I \models p \wedge q) = \text{uu}$) can be represented in two-valued logic as the satisfaction of the following formula according to the same I :

$$\neg(p = \text{tt} \wedge q = \text{tt}) \wedge \neg(p = \text{ff} \vee q = \text{ff}) \equiv (p = \text{uu} \wedge q \neq \text{ff}) \vee (q = \text{uu} \wedge p \neq \text{ff})$$

where $v \neq \text{tv}$ is a shorthand for $\neg(v = \text{tv})$.

Obviously, for two-valued variables the procedure just described is redundant and we will build two-valued formulas directly on variables in Vars.

2.1 Reactive Modules with Imperfect Information

We start by providing details on a notion of agent in the Simple Reactive Modules Language [57], and in particular its extension with imperfect information presented in [38]. In this context, agents are also known as *modules*.

DEFINITION 1 (RMI). *Given a finite set Vars of variables, a reactive module with imperfect information (RMI) is a tuple $m_a = \langle V_a, \text{Vis}_a, \text{update}_a \rangle$ such that*

- $V_a \subseteq \text{Vars}$ is the set of variables controlled by module m_a .
- $\text{Vis}_a \subseteq \text{Vars}$ is the set of variables visible to module m_a , where $\text{Vis}_a \cap V_a = \emptyset$.
- update_a is a finite set of update commands of the form:

$$\gamma ::= g \rightsquigarrow v_1 := \text{tv}_1, \dots, v_k := \text{tv}_k$$

where each $v_i \in V_a$ occur at most once in v_1, \dots, v_k , guard g is a Boolean formula over $\text{Vars}_a = V_a \cup \text{Vis}_a$, and $\text{tv}_i \in \text{Dom}(v_i)$, i.e., a value in the domain of variable v_i .

We denote with $g(\gamma)$ and $\text{asg}(\gamma)$ the guard (here, g) and assignment of γ , respectively. Moreover, we denote with $\text{Vars}(\phi)$ the set of variables occurring in formula ϕ .

The intuitive reading of a guarded command $g \rightsquigarrow \text{asg}$ is: if guard g is true, then the agent executes assignment asg . We can introduce a skip action $\text{skip} ::= \text{tt} \rightsquigarrow \epsilon$, with trivial guard tt and empty assignment ϵ .

Example 1. To fully understand the definition, we can consider the following two reactive modules:

- $m_a = \langle V_a, \text{Vis}_a, \text{update}_a \rangle$ such that
 - $V_a = \{v_1\}$
 - $\text{Vis}_a = \{v_2\}$

– update_a :

$$\begin{aligned}\gamma_a^1 &::= v_1 = \text{tt} \wedge v_2 = \text{ff} & \rightsquigarrow v_1 := \text{tt} \\ \gamma_a^2 &::= v_1 = \text{tt} \wedge v_2 = \text{ff} & \rightsquigarrow v_1 := \text{ff} \\ \text{skip}_a &::= \text{tt} & \rightsquigarrow \epsilon\end{aligned}$$

• $m_b = \langle V_b, \text{Vis}_b, \text{update}_b \rangle$ such that

- $V_b = \{v_2\}$
- $\text{Vis}_b = \emptyset$
- update_b :

$$\begin{aligned}\gamma_b^1 &::= v_2 = \text{ff} & \rightsquigarrow v_2 := \text{tt} \\ \gamma_b^2 &::= v_2 = \text{ff} & \rightsquigarrow v_2 := \text{ff} \\ \text{skip}_b &::= \text{tt} & \rightsquigarrow \epsilon\end{aligned}$$

That is, the first module m_a owns the variable v_1 and can see the variable v_2 , while the module m_b owns the variable v_2 but cannot see the variable v_1 . Additionally, both modules m_a and m_b have three update commands. In particular, module m_a can modify the value of variable v_1 when variable v_1 is true and variable v_2 is false; in all other cases, it does nothing (skip command). Meanwhile, module m_b can modify the value of variable v_2 when variable v_2 is false; in all other cases, it does nothing (skip command).

DEFINITION 2 (RMI ARENA). An RMI arena is defined to be an $(|Ag| + 4)$ -tuple $\mathcal{A} = \langle Ag, \text{Vars}, AP, \Phi_I, m_1, \dots, m_{|Ag|} \rangle$, where:

- Ag is the set of indexes for agents.
- Vars is the set of variables and $AP \subseteq \text{Vars}$ is the set of atomic propositions.
- Φ_I is a formula on AP specifying the set of initial states.
- For every $a \in Ag$, $m_a = \langle V_a, \text{Vis}_a, \text{update}_a \rangle$ is an RMI.

Hereafter, we assume that control is *exclusive*: for any two distinct agents a and b , $V_a \cap V_b = \emptyset$, i.e., the sets of controlled atoms are disjoint. Then, we often talk about the *owner* $\text{own}(v) \in Ag$ of a variable $v \in \text{Vars}$.

Example 2. We can define an RMI $\mathcal{A} = \langle Ag, \text{Vars}, AP, \Phi_I, m_a, m_b \rangle$, where:

- $Ag = \{a, b\}$;
- $\text{Vars} = AP = \{v_1, v_2\}$;
- $\Phi_I = v_1 = \text{tt} \wedge v_2 = \text{ff}$;
- $m_a = \langle V_a, \text{Vis}_a, \text{update}_a \rangle$ and $m_b = \langle V_b, \text{Vis}_b, \text{update}_b \rangle$ are the RMI defined in Example 1.

Thus, in this example we have two agents, a and b , two atomic propositions v_1 and v_2 (the set of variables and the atomic propositions coincide in this case), and the set of initial states is a singleton in which v_1 is true and v_2 is false.

2.2 Alternating-time Temporal Logic

To reason about the strategic abilities of agents in RMI arenas, we make use of the Alternating-time Temporal Logic ATL^* , whose syntax is defined as follows.

DEFINITION 3 (ATL^*). *State (φ) and path (ψ) formulas in ATL^* are defined inductively as follows, for $q \in AP$ and $A \subseteq Ag$.*

$$\begin{aligned}\varphi &::= q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A \rangle\rangle\psi \\ \psi &::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid X\psi \mid \psi U \psi\end{aligned}$$

The formulas of ATL^* are all and only the state formulas.

The strategy operator $\langle\langle A \rangle\rangle$ is read as “coalition A of agents can achieve ...”. The meaning of LTL operators ‘next’ X and ‘until’ U is standard. The Boolean operators \vee , \rightarrow , temporal operator ‘release’ R , and dual operator ‘unavoidably’ $\llbracket A \rrbracket$, are defined as usual.

Example 3. Given the RMI arena of Example 2, we can specify a simple ATL^* property like $\varphi = \langle\langle \{a, b\} \rangle\rangle X v_2$. Thus, we can check whether agents a and b have a collective strategy to satisfy in the next state the atomic proposition v_2 .

2.3 RMI arenas as Concurrent Game Structures

In order to interpret the language of ATL^* on RMI arenas, we show how to embed them into concurrent game structures with imperfect information [38], the semantics of choice for ATL^* .

DEFINITION 4 (EMBEDDING OF RMI ARENAS INTO ICGS). *An RMI arena $\mathcal{A} = \langle Ag, Vars, AP, \Phi_I, m_1, \dots, m_{|Ag|} \rangle$ is associated with the concurrent game structure $M_{\mathcal{A}} = \langle S, S_0, \{Act_a\}_{a \in Ag}, \pi, \{\sim_a\}_{a \in Ag}, P, \tau \rangle$, defined as follows:*

- $S = \{s \in Eval(Vars)\}$ is the set of states, where $Eval(Vars)$ is the set of all evaluations $I : Vars \rightarrow Dom(Vars)$ for the variables in $Vars$ into their respective domains.
- The set S_0 of initial states is the set of states that satisfy formula Φ_I .
- For every agent $a \in Ag$, $Act_a = update_a$ is the set of actions of agent a .
Then, $Act = \bigcup_{a \in Ag} Act_a$ is the set of all actions, and $ACT = \prod_{a \in Ag} Act_a$ is the set of all joint actions.
- The labelling function $\pi : S \times AP \rightarrow \{tt, ff\}$ labels each state with the atoms that are true in it, i.e., $\pi(s, q) = tt$ iff $s(q) = tt$.
- The indistinguishability relation is defined so that for every $s, s' \in S$, $s \sim_a s'$ iff for each $v \in Vars_a$, $s(v) = s'(v)$. Clearly, \sim_a is an equivalence relation.
- For every state $s \in S$ and agent $a \in Ag$, the protocol function $P : S \times Ag \rightarrow (2^{Act} \setminus \emptyset)$ returns the set $P(s, a)$ of enabled update-commands γ such that $s \models g(\gamma)$.
In particular, if $s \sim_a s'$ then $s \models g(\gamma)$ iff $s' \models g(\gamma)$, hence, $P(s, a) = P(s', a)$.
- The transition function $\tau : S \times ACT \rightarrow S$ is such that a transition $\tau(s, (\gamma_1, \dots, \gamma_n)) = s'$ holds iff
 - for every $a \in Ag$, $\gamma_a \in P(s, a)$;
 - for every $v \in Vars$ and $own(v) \in Ag$, $\pi(s', v) = tv$ iff either $asg(\gamma_{own(v)})$ contains an assignment $v := tv$ or, $\pi(s, v) = tv$ and $asg(\gamma_{own(v)})$ does not contain an assignment $v := tv'$, where $tv, tv' \in Dom(v)$ and $tv' \neq tv$.

The iCGS associated with an RMI arena describes the interactions of the agents in set Ag . Each agent a performs the actions in $Act_a = update_a$ according to the protocol function P . By acting synchronously, the agents brings about

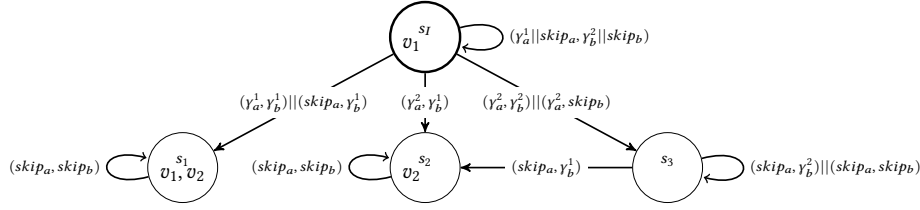


Fig. 1. An iCGS for the RMI arena of Example 2. Note that, $||$ is used when there are multiple possibilities to activate the same transition from a state. For simplicity of writing, we used this operator on tuples or single actions.

change in the system according to the transition function τ . Further, every agent a has *imperfect information* about the global state of the iCGS, and in every state s she considers any state $s' \sim_a s$ as (epistemically) possible [30, 43]. Equivalently, the information set of an agent a can be defined as the equivalence class $[s]_a = \{s' \in S \mid s' \sim_a s\}$. Agents have *perfect information* whenever every equivalence relation \sim_a is the identity relation, that is, all equivalence classes $[s]_a$ are singletons.

REMARK 1. Note that the size of the iCGS associated with an arena \mathcal{A} is exponential in the number of states of \mathcal{A} .

Example 4. Given the RMI arena of Example 2, we can generate the iCGS depicted in Figure 1. Here, we have four states $S = \{s_I, s_1, s_2, s_3\}$, where s_I represents the initial state. The actions that label the transitions are the possible combinations of the enabled update commands for a and b . Since agent b cannot see variable v_1 , we have $s_1 \sim_b s_2$ and $s_I \sim_b s_3$. In fact, b can only distinguish states that differ in the truth value of v_2 .

2.4 ATL Semantics on RMI arenas

The interpretation of ATL^* with imperfect information and perfect recall [43] is defined on RMI arenas by using the associated iCGS. Given an RMI arena, a *trace* is a (finite or infinite) sequence $tr = s_1 \vec{\alpha}_1 s_2 \vec{\alpha}_2 \dots$ such that for every i , $s_{i+1} = \tau(s_i, \vec{\alpha}_i)$. The length of the trace is the maximal index of an action $\vec{\alpha}_i$ in the sequence. A *path* is then a trace from which the actions are ignored, i.e., $p = s_1 s_2 \dots$. Also a finite, non-empty path $h \in S^+$ is called a *history*. Hereafter, we extend the indistinguishability relation \sim_a to histories as follows: for every $h, h' \in S^+$, $h \sim_a h'$ iff $|h| = |h'|$ and for every $i \leq |h|$, $h_i \sim_a h'_i$.

DEFINITION 5 (STRATEGY). A uniform, memoryful strategy for agent $a \in Ag$ is a function $f_a : S^+ \rightarrow Act_a$ such that for all histories $h, h' \in S^+$, we have:

- (i) $f_a(h) \in P(\text{last}(h), a)$;
- (ii) if $h \sim_a h'$ then $f_a(h) = f_a(h')$.

Hereafter we consider the *objective* and *subjective* interpretations of ATL^* , which are its most popular semantics [43]. Formally, given a *joint strategy* $F_A = \{f_a \mid a \in A\}$ for coalition $A \subseteq Ag$, and history $h \in S^+$, let $out_{obj}(h, F_A)$ be the set of all infinite paths p starting from h and consistent with F_A , that is, $p \in out_{obj}(h, F_A)$ iff $p_{\leq |h|} = h$, and for all $i \geq |h|$, $p_{i+1} = \tau(p_i, \vec{\alpha}_i)$, where for all $a \in A$, $\alpha_a = f_a(p_{\leq i})$. Then, $out_{sub}(h, F_A) = \bigcup_{h' \sim_a h, a \in A} out_{obj}(h', F_A)$, that is, it is the set of all infinite paths p whose initial segment is indistinguishable from h , for some $a \in A$, and consistent with F_A .

We now assign a meaning to ATL^* formulas on iCGSs.

DEFINITION 6 (SATISFACTION). *The satisfaction relation \models_x for $x \in \{obj, sub\}$, an iCGS \mathbb{M} , state s , infinite path p , state formula φ , and path formula ψ is defined as follows:*

$$\begin{aligned}
(\mathbb{M}, s) \models_x q & \quad \text{iff } \pi(s, q) = \text{tt} \\
(\mathbb{M}, s) \models_x \neg \varphi & \quad \text{iff } (\mathbb{M}, s) \not\models_x \varphi \\
(\mathbb{M}, s) \models_x \varphi \wedge \varphi' & \quad \text{iff } (\mathbb{M}, s) \models_x \varphi \text{ and } (\mathbb{M}, s) \models_x \varphi' \\
(\mathbb{M}, s) \models_x \langle\langle A \rangle\rangle \psi & \quad \text{iff for some strategy } F_A, \text{ for all infinite paths } p' \in \text{out}_x(s, F_A), (\mathbb{M}, p') \models_x \psi \\
(\mathbb{M}, p) \models_x \varphi & \quad \text{iff } (\mathbb{M}, p_1) \models_x \varphi \\
(\mathbb{M}, p) \models_x \neg \psi & \quad \text{iff } (\mathbb{M}, p) \not\models_x \psi \\
(\mathbb{M}, p) \models_x \psi \wedge \psi' & \quad \text{iff } (\mathbb{M}, p) \models_x \psi \text{ and } (\mathbb{M}, p) \models_x \psi' \\
(\mathbb{M}, p) \models_x X\psi & \quad \text{iff } (\mathbb{M}, p_{\geq 2}) \models_x \psi \\
(\mathbb{M}, p) \models_x \psi_1 U \psi_2 & \quad \text{iff for some } j \geq 1, (\mathbb{M}, p_{\geq j}) \models_x \psi_2, \text{ and for all } 1 \leq k < j, (\mathbb{M}, p_{\geq k}) \models_x \psi_1
\end{aligned}$$

The semantics of operators ‘unavoidably’ $\llbracket A \rrbracket$, ‘eventually’ F , ‘globally’ G , and ‘release’ R can be introduced as standard. Hereafter, we remove the subscripts *obj*, *sub* whenever inconsequential or clear from the context.

A formula φ is *true in an iCGS \mathbb{M}* , or $\mathbb{M} \models \varphi$, iff for all initial states $s_0 \in S_0$, $(\mathbb{M}, s_0) \models \varphi$. Finally, a formula φ is *true in an RMI arena \mathcal{A}* , or $\mathcal{A} \models \varphi$, iff φ is true in the associated iCGS $\mathbb{M}_{\mathcal{A}}$.

Notice that the *subjective* interpretation of ATL^* (with perfect recall), whereby operator $\langle\langle A \rangle\rangle$ is evaluated on all paths $p \in \text{out}(s, F_A)$, epistemically consistent with the current state s , is in contrast with the *objective* interpretation, which only considers paths actually starting from the current state. The subjective interpretation of ATL^* allows us to talk about the strategic abilities of agents as depending on their knowledge, which is essential in the analysis of security protocols. We illustrate this point in Section 4.

Hereafter we tackle the following major decision problem.

DEFINITION 7 (MODEL CHECKING PROBLEM). *The model checking problem for RMI arenas has as inputs an RMI arena \mathcal{A} and an ATL^* formula ϕ and asks whether $\mathcal{A} \models \phi$.*

The following theorem is an immediate consequence of the the undecidability of ATL with imperfect information, under the assumption of perfect recall [4, 29]:

THEOREM 5. *The model checking problem for ATL on RMI arenas is undecidable.*

In the rest of the paper, we will use the following usual notations from [30], without introducing commonplace notions such as common knowledge and distributed knowledge:

- $\sim_A^E = \bigcup_{a \in A} \sim_a$ denotes the *collective knowledge relation* for coalition A .
- $\sim_A^C = (\sim_A^E)^*$ denotes the *common knowledge relation* for coalition A , where $*$ is the transitive closure.
- $\sim_A^D = \bigcap_{a \in A} \sim_a$ denotes the *distributed knowledge relation* for coalition A .

All these notations are overloaded over states and histories: let \sim be any of \sim_a , \sim_A^E , \sim_A^C , or \sim_A^D , for $a \in Ag$ and $A \subseteq Ag$. Then, for all histories h, h' , we set $h \sim h'$ iff (i) $|h| = |h'|$, and (ii) for every $j \leq |h|$, $h_j \sim h'_j$. Furthermore, we denote with $C_A(h)$ the *common knowledge neighbourhood* of history $h \in \text{Hist}(\mathbb{M})$ for coalition $A \subseteq Ag$. That is, $C_A(h) = \{h' \mid h \sim_A^C h'\}$.

3 VISIBILITY CONTROL

The scope of this section is twofold. Firstly, we present a class of reactive modules in which the visibility control of atoms is explicit, in the sense that specific commands may allow each variable to be declared as visible or invisible to some agent. The relevance of this visibility control mechanism will be apparent in the Terrorist Fraud case study, that we present in Section 4.

Secondly, we identify a subclass of reactive modules with visibility control which accommodate a decidable model-checking problem for ATL^* , under the assumption of imperfect knowledge and perfect recall. This subclass is inspired by and generalizes systems without information forks [33]. We also give two semantic counterparts of the relevant property, the *strategic distributed knowledge* and the *A-cast* properties, defined on iCGS, and state the decidability of a subclass of the model-checking problem, namely for formulas which utilize coalition operators only for coalitions that have strategic distributed knowledge. The role of the *A-cast* property is to make the connection between visibility control with no information forks and strategic distributed knowledge.

3.1 Reactive modules with visibility control

DEFINITION 8 (RMI ARENA WITH VISIBILITY CONTROL). *An RMI arena with visibility control is a tuple $\mathcal{A} = \langle Ag, Vars, AP, \Phi_I, m_1, \dots, m_{|Ag|} \rangle$, where:*

- *Ag, Vars, and AP are defined as for RMI arenas in Def. 2.*
We denote with $VA = Vars \setminus AP$ the set of visibility variables and for each $v \in VA$, $Dom(v) = \{tt, ff, \uparrow\}$, where \uparrow is the undefined value. Furthermore, for each $a \in Ag$, we define a mapping $vis_a : V_a \times Ag \rightarrow VA$ and denote $vis_a(v, b)$ as $v_{a,b}$.
- *Each disjunct d_i of formula Φ_I needs to satisfy the following constraint: for every variable $v_{a,b} \in VA \cap Vars(d_i)$, if $v_{a,b} = tv$ appears in d_i , for $tv \in \{tt, ff\}$, then $v = tv$, i.e., variable v has to have the same truth value of $v_{a,b}$, intuitively.*
- *Every module $m_i = \langle V_a, Vis_a, update_a \rangle$ has the following additional constraints:*
 - $\{v_{a,b} \in VA \mid b \neq a\} \subseteq V_a$.
 - $Vis_a = \{v_{b,a} \in VA \mid b \neq a\}$, i.e., the set of visibility variables for a .
 - *Each command in $update_a$ is of the form:*

$$\begin{aligned} \gamma ::= & \quad g \quad \rightsquigarrow \quad v_1 := tv_1, \dots, v_k := tv_k \\ & \quad v_{a,b_1} := tv_{k+1}, \dots, v_{a,b_m} := tv_{k+m} \end{aligned}$$

where $v_i \in V_a$, $v_{a,b_i} \in VA$, and $b_1, \dots, b_m \in Ag \setminus \{a\}$.

Let us give some intuition around Definition 8. In an RMI arena with visibility control, each agent a literally has actions that not only update the values of its own variables v_a , but also these actions by a set some of these variables in V_a to be visible or invisible to other agents b (i.e., it modulates the set Vis_b of variables). So, the $update_a$ command says that if guard g holds (which is a boolean condition on all a 's variables), then agent a will respectively set values tv_1, \dots, tv_k to its variables v_1, \dots, v_k , but also set/reset the visibility to “on”/“off” (tv_{k+1}, \dots) for variables of its own for other agents b_i . As per previous definitions, these latter variables denoted v_{a,b_i} are still variables in V_a but there can be observed by other agents, i.e., they are also in VA .

Further, we require commands in our RMI arenas with visibility control to be *consistent*.

DEFINITION 9 (CONSISTENT COMMAND). A command γ is consistent iff whenever $v_{a,b_i} = \text{tv}_{k+i}$ appears in $\text{asg}(\gamma)$, for $\text{tv}_{k+i} \in \{\text{tt}, \text{ff}\}$, then either $v := \text{tv}_{k+i}$ appears in $\text{asg}(\gamma)$ as well, or $g \models (v = \text{tv}_{k+i})$, i.e., either the variable v associated with the visibility variable v_{a,b_i} is assigned the same truth value, or v has already the same truth value of v_{a,b_i} .

Intuitively, by the above definition we can make explicit the ability for each agent to make visible or invisible their own variables for some other agent. Given a variable $v_{a,b}$, agent a is the owner of v and if a makes $v_{a,b} = \text{tt}$ (resp., $v_{a,b} = \text{ff}$) then agent b can see the variable v and its truth value. The latter follows by the second and third items of Def. 8, whereby v has the same truth value as $v_{a,b}$ in case $v_{a,b} = \text{tt}$ or $v_{a,b} = \text{ff}$. Otherwise $v_{a,b} = \uparrow$ and by consequence agent b can not see variable v .

Example 6. By following Example 2, we can define an RMI with visibility control $\mathcal{A} = \langle \text{Ag}, \text{Vars}, \text{AP}, \Phi_I, m_a, m_b \rangle$, where:

- $\text{Ag} = \{a, b\}$;
- $\text{Vars} = \{v_1, v_2, \text{vis}_a(v_1, b), \text{vis}_b(v_2, a)\}$;
- $\text{AP} = \{v_1, v_2\}$;
- $\Phi_I = v_1 = \text{tt} \wedge v_2 = \text{ff} \wedge \text{vis}_a(v_1, b) = \uparrow \wedge \text{vis}_b(v_2, a) = \text{ff}$;
- $m_a = \langle V_a, \text{Vis}_a, \text{update}_a \rangle$ such that
 - $V_a = \{v_1, \text{vis}_a(v_1, b)\}$
 - $\text{Vis}_a = \{\text{vis}_b(v_2, a)\}$
 - update_a :

$$\begin{aligned}
 \gamma_a^1 &::= v_1 = \text{tt} \wedge \text{vis}_a(v_1, b) = \uparrow \wedge \text{vis}_b(v_2, a) = \text{ff} \rightsquigarrow v_1 := \text{tt}, \text{vis}_a(v_1, b) := \uparrow \\
 \gamma_a^2 &::= v_1 = \text{tt} \wedge \text{vis}_a(v_1, b) = \uparrow \wedge \text{vis}_b(v_2, a) = \text{ff} \rightsquigarrow v_1 := \text{ff}, \text{vis}_a(v_1, b) := \uparrow \\
 \gamma_a^3 &::= v_1 = \text{tt} \wedge \text{vis}_a(v_1, b) = \uparrow \wedge \text{vis}_b(v_2, a) = \text{ff} \rightsquigarrow \text{vis}_a(v_1, b) := \text{tt} \\
 \gamma_a^4 &::= v_1 = \text{tt} \wedge \text{vis}_a(v_1, b) = \uparrow \wedge \text{vis}_b(v_2, a) = \uparrow \rightsquigarrow v_1 = \text{ff} \wedge \text{vis}_a(v_1, b) := \text{ff} \\
 \text{skip}_a &::= \text{tt} \rightsquigarrow \epsilon
 \end{aligned}$$

- $m_b = \langle V_b, \text{Vis}_b, \text{update}_b \rangle$ such that
 - $V_b = \{v_2, \text{vis}_b(v_2, a)\}$
 - $\text{Vis}_b = \{\text{vis}_a(v_1, b)\}$
 - update_b :

$$\begin{aligned}
 \gamma_b^1 &::= v_2 = \text{ff} \wedge \text{vis}_a(v_1, b) = \uparrow \wedge \text{vis}_b(v_2, a) = \text{ff} \rightsquigarrow v_2 := \text{tt}, \text{vis}_b(v_2, a) = \text{tt} \\
 \gamma_b^2 &::= v_2 = \text{ff} \wedge \text{vis}_a(v_1, b) = \uparrow \wedge \text{vis}_b(v_2, a) = \text{ff} \rightsquigarrow v_2 := \text{ff}, \text{vis}_b(v_2, a) = \text{ff} \\
 \text{skip}_b &::= \text{tt} \rightsquigarrow \epsilon
 \end{aligned}$$

Thus, with RMI with visibility control, we add additional variables that enable agents to modify the visibility of the atoms they control. Therefore, following our running example, agent a (resp. b) can make variable v_1 (resp. v_2) visible or invisible to agent b (resp. a) through the variable $\text{vis}_a(v_1, b)$ (resp. $\text{vis}_b(v_2, a)$). Note that this capability must then adhere to the agent's ability through the update commands. In fact, assuming the update commands defined in the respective modules, it can be observed that the ability to change the visibility of an atom has actually been assigned only to agent a via update commands γ_a^1 and γ_a^2 .

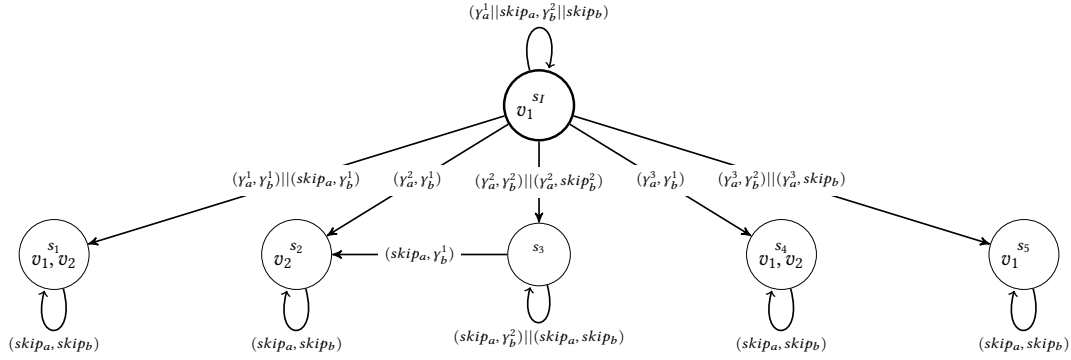


Fig. 2. An iCGS for the RMI arena with visibility control of Example 6. Note that, \parallel is used when there are multiple possibilities to activate the same transition from a state. For simplicity of writing, we used this operator on tuples or single actions.

Given the above RMI arena with visibility control, we can construct the iCGS depicted in Figure 2. In this case, we have six states: an initial state s_I and five additional states s_1, \dots, s_5 . Notice that we do not have all the possible combinations of the update commands for a and b since the fourth update command γ_a^4 of a is not available in the initial state (and also in the other states) because the guard of γ_a^4 is not satisfied. For the equivalence relations, we follow the same reasoning as in Example 4, that is $s_1 \sim_b s_2$ and $s_I \sim_b s_3$.

3.2 iCGS with Coalition Broadcast

In this section, we introduce our notions of *strategic distributed knowledge* and *A-cast iCGS*, and we state our decidability result. The first notion introduced, *A-cast*, models situations in which any information which an agent from outside a coalition A wants to send to some member of the coalition is broadcasted to all the members of A . The formal statement is the following:

DEFINITION 10 (COALITION BROADCAST). *Given an iCGSM and a coalition A , we say that M is A-cast if for any states $s_1, s_2, s'_1, s'_2 \in S$ and actions $\alpha_1, \alpha_2 \in ACT$, the following condition holds:*

If

- $s_1 \sim_A^D s_2$
- $\tau(s_1, \alpha_1) = s'_1$ and $\tau(s_2, \alpha_2) = s'_2$
- $\alpha_1|_A = \alpha_2|_A$
- $s'_1 \sim_A^E s'_2$,

then $s'_1 \sim_A^D s'_2$.

Note that this is a structural notion on iCGS and can be checked easily. Intuitively, by Def. 10, whenever a coalition A of agents has distributed knowledge, the agents in A perform some action, and in the resulting states everybody in coalition A knows some fact, then this fact is also distributed knowledge. This corresponds to the notion that distributed knowledge is preserved within the coalition.

Next, we give an example of an RMI, and its underlying CGS is an A-cast iCGS; it is easy to see the two agents a and b in the iCGS and their *update* commands define the states of this A-cast iCGS clearly. We do not draw it as a state-transition figure, since there are over 14 unwound states

Example 7. To fully understand the concept of A-cast, let's assume that our coalition A is composed of agents a and b and add an additional module $m_c = \langle V_c, Vis_c, update_c \rangle$ to our RMI with visibility control as follows:

- $V_c = \{v_3, vis_c(v_3, a), vis_c(v_3, b)\}$
- $Vis_c = \{vis_a(v_1, c), vis_b(v_2, c)\}$
- $update_c$:

$$\begin{aligned} \gamma_c^1 &::= v_3 = \text{tt} \rightsquigarrow vis_c(v_3, a) := \text{tt}, vis_c(v_3, b) := \uparrow \\ \gamma_c^2 &::= v_3 = \text{ff} \rightsquigarrow vis_c(v_3, a) := \uparrow, vis_c(v_3, b) := \text{ff} \\ skip_c &::= \text{tt} \rightsquigarrow \epsilon \end{aligned}$$

Note that the RMI with visibility control composed of the three modules m_a , m_b , and m_c generates an iCGS that is not $\{a, b\}$ -cast. In fact, depending on the value of the variable v_3 , the module m_c makes the variable visible to either module m_a or module m_b , thus not adhering to the concept of broadcast as defined in Definition 10.

An example of module m_c that respects the $\{a, b\}$ -cast property is the one with the following update commands:

$$\begin{aligned} \gamma_c^1 &::= v_3 = \text{tt} \rightsquigarrow vis_c(v_3, a) := \text{tt}, vis_c(v_3, b) := \text{tt} \\ \gamma_c^2 &::= v_3 = \text{ff} \rightsquigarrow vis_c(v_3, a) := \text{ff}, vis_c(v_3, b) := \text{ff} \\ \gamma_c^3 &::= \text{tt} \rightsquigarrow vis_c(v_3, a) := \uparrow, vis_c(v_3, b) := \uparrow \\ skip_c &::= \text{tt} \rightsquigarrow \epsilon \end{aligned}$$

The second notion we introduce here, which is strategic distributed knowledge inside a coalition A , is a technical property on strategies of A which allows a decidable model-checking problem. Intuitively, the members of A have strategic distributed knowledge if, for each joint strategy σ , the restriction of the common knowledge relation to histories compatible with σ is identical with the restriction of the distributed knowledge relation. In some sense, this can be seen as a generalization of the decidability of model-checking of ATL for coalitions whose strategies are based on distributed knowledge, as in [28].

DEFINITION 11 (STRATEGIC DISTRIBUTED KNOWLEDGE). *Given an iCGSM and a coalition $A \subseteq Ag$, we say that A has strategic distributed knowledge (DS-knowledge for short) in M if the following property holds:*

- *For every uniform strategy σ_A and histories $h_1, h_2 \in out(\sigma_A)$ such that $h_1 \sim_A^C h_2$, $h_1[1] = h_2[1]$ and $lst(h_1) = lst(h_2)$, we have that $h_1 \sim_A^D h_2$.*

The connection between the two notions is ensured by the following result:

LEMMA 1. *In any A-cast iCGSM, coalition A has DS-knowledge.*

PROOF. We actually prove the following stronger property: if $h_1, h_2 \in out(\sigma_A)$, $h_1[1] \sim_A^D h_2[1]$ and $h_1 \sim_A^C h_2$, then $h_1 \sim_A^D h_2$. This can be proved by induction on $|h_1| = |h_2|$. Note that the base case is trivial.

Assume now $|h_1| = |h_2| = n + 1$ and $h_1 \sim_A^C h_2$. We only prove that, when $h_1 \sim_A^E h_2$, we get $h_1 \sim_A^D h_2$ and then generalize by an inductive argument to conclude also for $h_1 \sim_A^C h_2$, since $\sim_A^C = (\sim_A^E)^*$. The assumption implies that $h_1[\leq n] \sim_A^E h_2[\leq n]$, which, by the inductive hypothesis, entails $h_1[\leq n] \sim_A^D h_2[\leq n]$. But then we have that $\sigma_A(h_1[\leq n]) = \sigma_A(h_2[\leq n])$ and since the assumption also entails $lst(h_1) \sim_A^E lst(h_2)$, the definition of A-cast can be applied to conclude that $lst(h_1) \sim_A^D lst(h_2)$. \square

More formally, suppose we have a family $C \in 2^{2^{A_g}}$. An ATL^* formula φ is called a C -formula if any coalition operator occurring in φ utilizes a coalition in C . The main result of this paper states the decidability of the model-checking problem for iCGS which satisfy $DS - knowledge$ for each coalition A in a family of coalitions C and for C -formulas.

THEOREM 8. *Given a sequence C of coalitions, the model-checking problem for C -formulas and for iCGS in which each coalition A in C has DS -knowledge is $2 - EXPTIME$ -complete.*

The proof of this theorem involves reducing the problem of checking whether each coalition A which has DS -knowledge may enforce an LTL formula ψ to the synthesis problem for formula ψ in a two-player, zero-sum game in which the "protagonist" has imperfect information. The DS -knowledge property ensures that the information sets for coalition A do not "grow unboundedly". One then only needs to treat the case of nested coalition operators, which can be done in the usual way, by identifying which states satisfy each subformula $\langle\langle A \rangle\rangle\psi$ where A is a coalition in C , then introducing new propositional atoms $p_{\langle\langle A \rangle\rangle\psi}$, which replace the subformulas $\langle\langle A \rangle\rangle\psi$, and then iterate.

Formally, the reduction to a finite game arena amounts to showing that the turn-based version $TB(\mathcal{G}, A)$ is bisimilar with a two-player, turn-based, zero-sum finite-state game arena with imperfect information which simulates the common knowledge inside coalition A , along each uniform strategy for A , and then using results from [25] to decide whether the protagonist of this turn-based game can enforce an LTL formula. In general, this two-player, turn-based, zero-sum game arena can be built for any iCGS, but this is an infinite-state game arena since the information sets corresponding with runs of length k grow unboundedly. This infinite information set construction appears already in [18]. The DS -knowledge property ensures that this infinite game structure is bisimilar with a finite game structure. The details of these semantic constructions are presented in Section 5.2.

3.3 RMI without Information Forks

In this section we identify a class of RMI arenas with visibility control which satisfy the A -cast property. We introduce the following notation: given an RMI arena $\mathcal{A} = \langle Ag, Vars, AP, \Phi_I, m_1, \dots, m_{|Ag|} \rangle$, an agent $c \in Ag$, an update command $\gamma \in \text{update}_c$ and a state $s \in Eval(Vars)$ such that $s \models g(\gamma)$, we denote $\gamma(s)$ the state resulting when c executes update γ while the other agents execute a *skip* action. More formally, $\gamma(s) = \tau(s, \alpha)$ where α is the tuple of updates with $\alpha_c = \gamma$ and $\alpha_a = \text{skip}$ for $a \neq c$.

Recall that, for two commands $\gamma_1, \gamma_2 \in \text{update}_c$, $s_1, s_2 \in Eval(Vars)$ and some $a \in Ag$, $\gamma_1(s_1) \not\sim_a \gamma_2(s_2)$ if there exists some $v \in V_c \cap Vis_a$ with $\gamma_1(s_1)(v) \neq \gamma_2(s_2)(v)$.

The following proposition gives a simple algorithm for checking that an RMI arena satisfies the A -cast property for some $A \subseteq Ag$. Recall that, given two states $s_1, s_2 \in S$, $s_1 \sim_A^D s_2$ iff for all $a \in A$ and $v \in V_a \cup Vis_a$ we have that $s_1(v) = s_2(v)$.

PROPOSITION 1. *Given an RMI arena $\mathcal{A} = \langle Ag, Vars, AP, \Phi_I, m_1, \dots, m_{|Ag|} \rangle$, the iCGS $M_{\mathcal{A}}$ associated with \mathcal{A} is A -cast iff the following property holds:*

- For every states $s_1, s_2 \in Eval(Vars)$ with $s_1 \sim_A^D s_2$, $c \notin A$, and commands $\gamma_1, \gamma_2 \in \text{update}_c$, if there exists some $a \in A$ such that $\gamma_1(s_1) \not\sim_a \gamma_2(s_2)$, then for all $b \in A$, $\gamma_1(s_1) \not\sim_b \gamma_2(s_2)$.

PROOF. Assume $s_1 \sim_A^D s_2$ and take some $\alpha_1, \alpha_2 \in ACT$, with $\alpha_1|_A = \alpha_2|_A$. For each $c \notin A$, denote $\gamma_i = \alpha_i|_c$. Assume $\tau(s_i, \alpha_i)$ is defined by the fact that $s_i \models g(\gamma_i)$. Then A -cast is equivalent with the fact that $\gamma_1(s_1) \sim_a \gamma_2(s_2)$ iff $\gamma_1(s_1) \sim_b \gamma_2(s_2)$ for any $a, b \in A$, which, in turn, is equivalent with the desired property. \square

DEFINITION 12 (INFORMATION FORKS). *Given an RMI with visibility control \mathcal{A} and coalition $A \in Ag$, we say that \mathcal{A} has no A -information forks (or no A -infoforks for short) if the following property holds:*

For any $c \notin A$, $v \in V_c$, $\gamma \in \text{update}_c$, $a, b \in A$, and $s \in \text{Eval}(\text{Vars})$, if $s \models g(\gamma)$ then $\gamma(s)(v_{c,a}) = \gamma(s)(v_{c,b})$.

LEMMA 2. *If \mathcal{A} has no A -infoforks then the associated iCGS $M_{\mathcal{A}}$ is A -cast.*

PROOF. Take $s_1, s_2 \in \text{Eval}(\text{Vars})$ with $s_1 \sim_A^D s_2$ and $\gamma_1, \gamma_2 \in \text{update}_c$ with $\gamma_1(s_1) \sim_a \gamma_2(s_2)$. This implies two things:

- (1) For any $a \in A$ and $v \in V_a$, $\gamma_1(s_1)(v) = \gamma_2(s_2)(v)$.
- (2) For any $v \in V_c$, and $v \in V_c$, $\gamma_1(s_1)(v_{c,a}) = \gamma_2(s_2)(v_{c,a})$.

By the hypothesis of the lemma, we further have that, for any $v \in V_c$ and any $a, b \in A$, $\gamma_i(s_i)(v_{c,a}) = \gamma_i(s_i)(v_{c,b})$, and hence, by combining with point 2 above, we get that $\gamma_1(s_1)(v_{c,a}) = \gamma_2(s_2)(v_{c,b})$. But this is sufficient for proving the property in Proposition 1, since $\gamma_1(s_1) \sim_a \gamma_2(s_2)$ iff $\gamma_1(s_1)(v) = \gamma_2(s_2)(v)$ for any $v \in V_a \cup \text{Vis}_a$. \square

Note that the converse of Lemma 2 does not hold. Consider an iCGS with three agents $Ag = a, b, c$ with $A = a, b$, in which we have some visibility atom \bar{v} and some update command $\gamma \in \text{update}_c$ with $\gamma(s)(\bar{v}_{c,a}) \neq \gamma(s)(\bar{v}_{c,b})$, but for all the other atoms $v \neq \bar{v}$ and all the update commands $\gamma' \in \text{update}_c$ with $\gamma' \neq \gamma$, the condition $\gamma'(s)(v_{c,a}) = \gamma'(s)(v_{c,b})$ holds. Furthermore, all commands in update_a never check the value of $\bar{v}_{c,a}$, and the same holds for all commands in update_b . Then the system is A -cast by an argument similar to the one in Lemma 2.

THEOREM 9. *Given a sequence C of coalitions, the model-checking problem for C -formulas and for RMI without A -infoforks for any $A \in C$ is decidable in $3 - \text{EXPTIME}$.*

PROOF. Given an RMI \mathcal{A} , we build the iCGS associated with \mathcal{A} , which requires an exponential explosion, and then apply Theorem 8. \square

REMARK 2. *We draw the attention of the reader to two important points about the constraints and relevance of visibility control in RMI. First, modifying some visibility variable might not be available as an atomic action, but only as part of an atomic action that modifies other visibility variables. In particular, some command of an agent a might, for example, make some variable v_1 visible to agent b while, at the same time, make some other variable v_2 invisible to b . Such commands are compatible with the property of absence of A -infoforks when both $a, b \in A$.*

On the other hand, even when the arena does not have A -infoforks for some coalition, when satisfying some ATL^ formula $\langle\langle A \rangle\rangle\varphi$, the strategies that coalition A would have to employ might involve keeping some variables owned by some agent $a \in A$ invisible to another agent $b \in A$.*

To see that, consider the following formula $\langle\langle a, b \rangle\rangle X(v_1 \wedge \neg K_a v_2)$, and recall that, with the subjective semantics, the knowledge operator can be expressed within ATL^ . Consider further that this objective must be achieved in some RMI in which a owns variable v_1 and b variable v_2 . Hence, the objective implies that b cannot employ some strategy in which v_2 becomes visible to a . An example of such an RMI is given below.*

Example 10. Consider the following RMI with two initial states which are indistinguishable for agent a :

- $Ag = \{a, b\}$;
- $\text{Vars} = \{v_1, v_2, \text{vis}_a(v_1, b), \text{vis}_b(v_2, a)\}$;
- $AP = \{v_1, v_2\}$;
- $\Phi_I = (v_1 = \text{ff} \wedge \text{vis}_b(v_2, a) = \uparrow)$;

- $m_a = \langle V_a, Vis_a, \text{update}_a \rangle$ with $V_a = \{v_1, vis_a(v_1, b)\}$, $Vis_a = \{vis_b(v_2, a)\}$ and update_a composed of the commands:

$$\gamma_a^1 ::= \text{tt} \rightsquigarrow v_1 := \text{tt}$$

$$\gamma_a^2 ::= \text{tt} \rightsquigarrow v_1 := \text{tt}$$

- $m_b = \langle V_b, Vis_b, \text{update}_b \rangle$ with $V_b = \{v_2, vis_b(v_2, a)\}$, $Vis_b = \{vis_a(v_1, b)\}$ and update_b composed of the commands:

$$\gamma_b^1 ::= v_2 = \text{tt} \rightsquigarrow vis_b(v_2, a) = \text{tt}$$

$$\gamma_b^2 ::= v_2 = \text{ff} \rightsquigarrow vis_b(v_2, a) = \text{ff}$$

$$\gamma_b^3 ::= \text{tt} \rightsquigarrow vis_b(v_2, a) = \uparrow$$

Clearly, this RMI does not have $\{a, b\}$ -infoforks since it vacuously satisfies Def. 12. In this RMI, the two agents may achieve the objective of enforcing the formula $X(v_1 \wedge \neg K_a v_2)$ in the initial states by choosing the joint action (γ_a^1, γ_b^1) . Note that b is forced to keep variable v_2 invisible from a in order to achieve this objective. Any other combination of actions does not achieve this objective from the initial states.

4 REACTIVE MODULES IN MODELLING SECURITY PROBLEMS

In security applications, parties can collude with attackers, secretly sharing information privately within such adversarial coalitions. We believe RMI *with visibility control* are therefore ideal to model such collusions in the context of security. We pick a specific example of such security-driven collusions, where the state of agents cannot encode the full history of the execution, thus providing an example pertinent to the proof in Section 5.2.

We start by recalling the setting of *terrorist-fraud attacks* [15] and a distance-bounding protocol by Bultel *et al.* [24], then give an RMI *with visibility control* specification for this protocol and an ATL formula expressing the terrorist-fraud security of this protocol. Finally we argue why the results from Section 5.2 are relevant for this case study.

To first exemplify this, we will recall the notion of *distance-bounding (DB)* protocol and coalition-based attacks therein.

4.1 Identity Schemes

Distance-bounding (DB) protocols [23] are identity schemes which can be summarised as follows: (1) via timed exchanges, a *prover* P demonstrates that they are physically situated within a bound from a *verifier* V ; (2) via these exchanges and the protocol itself, the prover also authenticates himself to the verifier.

DB protects against relaying (via time/distance measuring), but it opens for other attacks. In *terrorist-fraud (TF)* attacks, a prover P , who is malicious and far away from the honest verifier V , colludes with an adversary Adv (who is close to V) such that the coalition makes the verifier V believe that the far-away, dishonest P is close to V and acting legitimately. To fool V in this manner, the dishonest prover P could simply, for instance, give his ID card to adversary Adv , who could then impersonate P forever. But this is not a valid terrorist fraud. Instead, a valid terrorist-fraud attack is one where the dishonest P helps Adv in such a way that Adv can pass the protocol on P 's behalf without gaining any advantage for future authentication.

So, *TF-resistance* or *TF security* means any malicious, far-away prover P helping an adversary Adv show that P is close to a verifier V , leads to Adv getting an advantage. Much as this is counterintuitive, security in this case is by

deterrence. I.e., any prover who may try to help the attacker once, is dissuaded by the fact that this may lead to future, *unwanted* impersonations. This threat model, where help to the attacker is given up to a point, but not forever in the future is not ours: it is common in identification schemes and employed as such from the 80s to today [2].

Conversely, a *TF-attack* means that if there exists a coalition between a malicious, far-away prover P and an adversary Adv such that the verifier accepts Adv as P , then any memoryful strategy by Adv does not lead to the verifier accepting them as P , without P being active.

In Figure 3 below, we present a simplified version of the SPADE protocol [24]; this simplification is meaningful to our RMI modelling in this paper, in which we will not encode all the cryptographic verifications, constructions needed for the anonymisation of the prover, etc., but focus on the collusion and private-information sharing aspects.

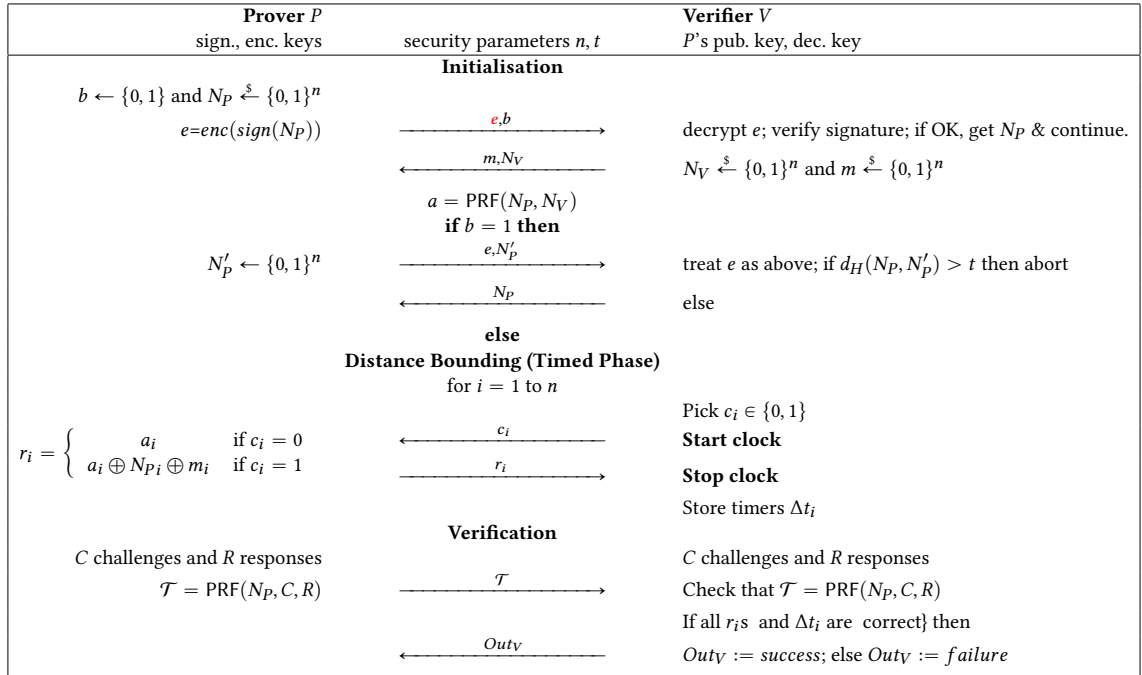


Fig. 3. A Description of the SPADE Protocol [24]

The SPADE protocol. The SPADE protocol, like other identification schemes, is a protocol in which a prover P is cryptographically identified/authenticated by a verifier V via series of challenge-response rounds, whereby the responses r_i in these rounds are based on the challenges as well as a secret m pre-established between the prover and the verifier. This pre-established secret m identifies the prover cryptographically. Moreover, the challenge-response rounds are timed by the verifier V and the responses are short (i.e., bits) such that this phase can happen fast and it ascertains that the prover P is physically close to V (as otherwise he will fail to answer within V 's hard-coded time-bound). So, such a protocol is not only an identification scheme but also an authenticated proximity-checking scheme.

We now explain the SPADE protocol in more technical detail and its TF-resistance mechanism; please refer to Figure 3 for the notations that follow. The prover P and the verifier V have the necessary cryptographic material for

V to authenticate P (i.e., verify their signature) and communicate securely N_P (i.e., the message e on Figure 3 is an encryption with V 's public key).

As per Figure 3, the protocol execution varies with the value of a bit b . Namely, if $b = 0$ then it is a “normal” execution, in which the prover is honest; if $b = 1$ then it is an “abnormal” execution, in which a terrorist-fraud attack may be attempted.

The normal/honest execution. We describe first the correct execution of the protocol. In the initialisation phase, P generates a *session key* in the form of a nonce² N_P , signs it, and encrypts it; the result e , along with the aforementioned bit b , is sent to V . The verifier V decrypts e and verifies the signature inside it; upon success, V generates and sends two pseudorandom bitstrings m, N_V . Both P and V pass N_V through a pseudorandom function (PRF) keyed on N_P , in order to yield a bitstring a .

Then, in the timed phase, P responds to n bit-challenges c_1, \dots, c_n and the verifier measures the duration of each exchange as $\Delta_1, \dots, \Delta_n$. If the i th challenge c_i is 0, then P 's i th response r_i is the i th bit of a , otherwise the i th response r_i is the result of XOR-ing the i th bits of a, m and N_P , respectively. The verifier accepts the prover/transaction if all answers are correct and in time, and a PRF-output τ keyed on the secret N_P on the current challenge and response bitstrings coincides with V 's view of the execution.

Note that, if run honestly ($b = 0$), the session-key N_P changes in each session; yet a man-in-the-middle attacker can learn on average $\frac{1}{4}$ of the bits of N_P (if the c_i s, a , N_P and m are sampled uniformly at random).

The terrorist-fraud-driven execution. First note, that if a MiM (man-in-the-middle) adversary replays an old e with $b = 0$, then only the V -issued, publicly-known N_V and m would change, and so half of the timed responses would be based on the old N_P and in this way, in albeit an exponential number of runs, this adversary could recover the old session key N_P .

Second, note that if a dishonest, far-away prover gives the list of all possible answers r_i for one run (based on one session key N_P), then the attacker gets the full corresponding N_P . (This is because the full response table leaks a , and m is passed in clear, so by simple XORs, the full N_P is leaked.) So, a far-away prover leaking the full response table for one protocol run, such that the colluding adversary Adv would pass one time, in fact leaks the session-key N_P for that very run (and the corresponding e would have been sent by this afar, dishonest prover so that Adv can pass the fast phase).

Third, note that if an attacker knows an old session-key N_P , then a replay of the e corresponding to that N_P (with $b = 1$) leads to a 100%-successful authentication. And, since recovering an old session-key N_P via a simple MiM-replay (i.e., with $b = 0$) takes exponential number of queries, if an adversary knows an old session-key N_P , it is because a dishonest prover leaked it or leaked a large part of it and the adversary guessed the rest. To dissuade a dishonest prover P from such leaking, the protocol has a built-in defence. If an attacker got (from a dishonest prover) a good part of N_P , i.e., knows an N'_P that differs from a given old N_P in no more than t bits, i.e., their Hamming distance³ $d_H(N_P, N'_P)$ is less than t , then this attacker/adversary can run the protocol with $b = 1$ and, as a punishment to the dishonest prover, V leaks the whole N_P . Now, by replaying the old e and knowing the N_P , this attacker can now impersonate the (dishonest) P forever.

So, in this protocol, in order to help an adversary pass the fast phase with a good probability, any far-away, dishonest prover P needs to leak (enough bits of) what is meant to be “just” an ephemeral session-key N_P . Yet, if that is so, then

²A nonce is a “number once used”, i.e., pseudorandom bitstring.

³The Hamming distance is the number of positions in two bitstrings of the same length that differ.

the adversary can turn against said prover and run the protocol in the ($b = 1$)-mode and then the adversary can get the full N_P impersonate this prover forever. This is how TF-resistance is built-in in the SPADE protocol.

In other words, to check TF-resistance (in SPADE), we will check the following intuitive, *ATL* formula:

$$\llbracket Adv \rrbracket G(\neg auth_without_help) \wedge \llbracket P, Adv \rrbracket ((\neg auth_without_help \wedge \neg auth_via_help) \ W \ (auth_via_help \wedge \llbracket Adv \rrbracket G auth_via_help))$$

In ATL^* , to check a TF attack (in SPADE), we will check the following formula:

$$\llbracket P, Adv \rrbracket F(auth_via_help \wedge \llbracket Adv \rrbracket G \neg auth_without_help),$$

These formulae are both explained later, on page 25.

4.2 Security of Identity Schemes Analysed via RMI with visibility control

We now give an RMI *with visibility control* that can be used to check if, in the SPADE protocol, TF-resistance expressed as the ATL formulae above, actually holds.

Modelling details. We consider the set $Ag = \{P, Adv, V\}$ of agents, i.e., prover P who can be honest as well as dishonest in different sessions/executions of the protocol, attacker Adv , and verifier V . The coalitions in RMI *with visibility control* are $\mathcal{A}_1 = \{P, Adv\}$ and $\mathcal{A}_2 = \{Adv\}$.

To check for TF-resistance in our RMI *with visibility control* modelling, it is sufficient to do the following:

- (1) Discard e , m , N_V as variables in the RMI *with visibility control*, because e , m , N_V are sent on a public channel, thus every time Adv or V know an N_P , they also know the corresponding e .
- (2) Encode that P is close/far to V without explicit distances/timing.
- (3) We do not need to encode several distance-bounding bit-exchanges, as only one bitstring exchange and its leakage/non-leakage are sufficient to model the corrupt/honest behaviour herein.
- (4) The above implies that the only aspects that the prover P can leak privately to Adv are the responses to the DB challenges and the nonce N_P .

We model n executions of the SPADE protocol under adversarial conditions. Objects indexed by j refer to SPADE's execution number $j \in \{1, 2, \dots, n\}$.

In one arbitrarily-picked execution denoted d , the prover is far from V and gives away his timed-phase responses to the attacker. In fact, in this session, the prover gives away his current N_P or the pre-calculated response to the challenge. Apart from such leakages, the prover P behaves honestly, as per the SPADE specification. The verifier V is also honest.

In our variable modelling, we use the following enumeration types:

- $nonces_1^j$ for the domain of prover-generated N_P s, in execution j ;
- $nonces_2$ for adversarially-generated N_P s.

To capture the pseudo-randomness of responses and N_P , responses generated by P and Adv will respectively range over disjoint sets: e.g., each value-set $nonces_1^j$ for P , for each execution j , is disjoint from $nonces_2$ for Adv . However, attacker Adv can get to see values in $nonces_1^d$ during collusion in an execution d . All domains contains a sorted *null* value, used to model an aberrant value in that domain.

Agents. We now define the agents in the RMI *with visibility control*.

Controlled Variables. The set $VC_P = \{P_far^j, P_collude^j : \text{bool}, P_Np^j : \text{nonces}_1^j, P_b^j : \{0, 1\}, P_r^{j,ch} \in r_1^j \mid j \leq n, ch \in ch_set\}$ contains the variables controlled⁴ by prover P : whether P is far from V in the j -th SPADE-execution, whether P is colluding in the j -th SPADE-execution, P 's nonce Np in the j -th execution, P 's bit b in the j -th execution, P 's responses r to the DB challenge in the j -th execution, one for each given challenge ch in ch_set that the verifier V may set.

Similarly, Adv 's controlled variables are $VC_{Adv} = \{Adv_Np^j : \text{nonces}_1^j \cup \text{nonces}_2, Adv_b^j : \{0, 1\}, Adv_r^{j,ch} : r_1^j \cup r_2 \mid j \leq n, ch \in ch_set\}$, where $r_1^j \cap r_2 = \emptyset$, $\text{nonces}_2 \cap \text{nonces}_1^j = \emptyset$, for each $j \in \{1, \dots, n\}$. These conditions state that, if chosen legitimately, the prover and the adversary do not chose the same nonce and cannot generate the same responses to the DB challenges (as they have different secret keys).

The set $VC_V = \{V_rec_{Np}^j : \text{nonces}_1^j \cup \text{nonces}_2 \cup \{\text{null}\}, finished^j, ok^j, V_b^j : \{0, 1\}, V_ch^j \in ch_set, V_r^{j,ch} : r_1^j, V_rec_r^{j,ch} : r_1^j \cup r_2 \cup \{\text{null}\} \mid j \leq n, ch \in ch_set\}$ contains the variables controlled by V . For each execution j , V will receive Nps as part of the challenge-response phase and will store these in $V_rec_{Np}^j$, V will know if the execution has finished and it was successful, V will have a bit V_b for the execution mode, V will send a challenge V_ch out in any session j , V compute responses V_r for any session j and any possible chosen challenge-value ch , V receives responses V_rec_r for any session j and any possible chosen challenge-value ch .

Visible Variables. The set $Vis_P = \{V_ch^j : ch_set \mid j \leq n\}$ meaning that the prover could receive a challenge from the verifier potentially in every session, hence the V_ch^j variables by V are visible to the prover.

The set $Vis_{Adv} = \{V_ch^j : ch_set, P_Np^j : \text{nonces}_1^j, P_b^j : \{0, 1\}, P_r^{j,ch} \in r_1^j \mid j \leq n, ch \in ch_set\}$ meaning that the adversary could receive a challenge from the verifier potentially in every session, hence the V_ch^j variables by V are visible to the adversary, and the adversary can also see the prover's bits b , as well as nonces Np and DB-responses r , potentially in any session in case the prover is corrupt in that session.

The set $Vis_V = \{Adv_r^j \in r_2 \mid j \leq n\} \cup \{P_r^{j,ch} : r_1^j \mid j \leq n\} \cup \{P_Np^j : \text{nonces}_1^j \mid j \leq n\} \cup \{Adv_Np^j : \text{nonces}_2 \mid j \leq n\} \cup \{Adv_b^j : \{0, 1\} \mid j \leq n\}$ meaning that the verifier can see the DB responses from the adversary and/or the prover potentially in every session j of the execution of SPADE, that the verifier gets to see the Np nonce either from the prover or the adversary in each session j , and that the verifier gets to see the bits b as per what the adversary lets then get through from the prover (flipped or unaltered) or sets himself.

Notation: To simplify the explanations that follow, we write:

- $val(x)$ for the value taken by a variable x ;
- P_y, V_y, \dots , for variable y in the prover and variable y in the verifier, etc;
- $vis(y, X) := \text{tt/ff}$ for variable y having its visibility changed, to true/false, for agent X ;
- $val(x \text{ in } vis(x, \cdot))$ for the value of variable x inside a visibility construct.

In the initial states, we implicitly assume that all visibility is set to false, unless we explicitly set it to true.

Initial States. For simplicity, we describe the initial states w.r.t. each agent. These initial states, as we will see below, can be expressed as propositional formulas, but we describe them in terms of variables' (in)equalities.

Prover's Initial States. We start with describing some of the possible states of the prover. In fact, we develop on the initial states of the prover when he colludes with the adversary, i.e., $P_collude^d = \text{tt}$, in some arbitrarily-picked execution d . We model the rest of the executions as arbitrary (i.e., could be collusive or not).

⁴Variables controlled by a vCGS agent b are normally written V_b , but here we write VC_b to avoid clashing with other notations in this protocols, i.e., the verifier.

What is not set specifically in the initial states (as per the below) can, clearly, be chosen non-deterministically as part of the first command. Via the initial states, we model that P pre-calculates the responses for all possible challenges, in each SPADE-execution j (i.e., $P_{r^{j,ch}} = val^{j,ch}$ in all the initial states below).

Setting 1 of Collusion. In real-life, this equates to a collusion whereby a malicious P preempts the illicit authentication by one execution. That means the prover is close-by the verifier and will run the full protocol, after which will leak the N_P of this session to the adversary, so that the latter impersonates him thereafter.

One possible initial state of the prover, which models this setting 1 of collusion, is as follows:

$$\begin{aligned} \bullet \text{ the prover: } & P_{far^d} = \text{ff}, P_{collude^d} = \text{tt}; P_{b^d} = 0; \quad P_{collude^k} = \text{ff}; P_{b^k} = 0 \\ & P_{r^{j,ch}} = val^{j,ch}; vis(P_{b^j}, Adv) = \text{tt}, \quad vis(P_{N_P^d}, V) = \text{tt} \end{aligned}$$

for each $j \in \{1, \dots, n\}$, for each $ch \in ch_set$, for d arbitrarily picked in $\{1, \dots, n\}$, for each $k \in \{1, \dots, n\}$, $k \neq d$, with $val^{j,ch}$ in $enum_1^j$.

In this initial state, the devious P is close-by V and P will answer in session d , that is, P will run session d with $b = 0$ (i.e., as P). So, P sends his nonce to V (i.e., $vis(P_{N_P^d}, V) = \text{tt}$).

Note that the adversary Adv cannot answer correctly in this session d , as P 's N_P is unknown to Adv . But in this collusion setting, we will force that right after session d , the prover leaks N_P to Adv , so Adv can play on $b = 1$ thereafter.

Setting 2 of Collusion. In real-life, this equates to a collusion whereby a malicious P asks for immediate illicit authentication from Adv . In this setting, in session d , it does not matter therefore if P is far or close (i.e., P_{far^d} is not set), but P_{b^d} has to be set to 0 so that the timed phase of session d is executed (by someone, be it P or Adv). If P is far, then for this phase to be successful, it will have to be run by Adv (i.e., Adv will send the answers to V 's challenges); if P is close, then the timed phase in session d can be run by either P or Adv (i.e., both transitions are possible).

One possible initial state of the prover, which models this setting 2 of collusion, is as follows:

$$\begin{aligned} \bullet \text{ the prover: } & P_{collude^d} = \text{tt}; \quad P_{b^d} = 0; P_{collude^k} = \text{ff}; P_{b^k} = 0; \\ & P_{r^{j,ch}} = val^{j,ch}; \quad vis(P_{N_P^d}, Adv) = \text{tt}, vis(P_{b^j}, Adv) = \text{tt} \\ & vis(P_{N_P^d}, V) = \text{tt}. \end{aligned}$$

In this initial state, the devious prover P reveals N_P (in real life e and N_P) to the adversary before the session starts. Yet, this session can be run with b^d remaining 0 all the way to the verifier⁵; to this end, P is releasing his nonce N_P to the verifier (i.e., $vis(P_{N_P^d}, V) = \text{tt}$).

After session d , Adv can also successfully run all future sessions with $b = 1$ (with no timed phase), just based on this learnt N_P .

Finally, one other possible initial state of the prover, also modelling setting 2 of collusion, is as follows:

$$\begin{aligned} \bullet \text{ the prover: } & P_{collude^d} = \text{tt}; \quad P_{b^d} = 0; P_{collude^k} = \text{ff}; P_{b^k} = 0; \\ & P_{r^{j,ch}} = val^{j,ch}; \quad \wedge_{ch \in ch_set} vis(P_{r^{d,ch}}, Adv) = \text{tt}, vis(P_{b^j}, Adv) = \text{tt}, \\ & vis(P_{N_P^d}, V) = \text{tt}. \end{aligned}$$

The above initial state means that, in the “malign” SPADE-execution d , P colludes with Adv and P reveals to the adversary the response-table $\cup_{ch \in ch_set} P_{r^{d,ch}}$ for the fast phase, so the adversary can reply to any possible challenge

⁵We will see via the update commands that Adv can flip the bit b^d or leave it as is b^d .

in this session d , instead of P . Yet, as per the above, this session can be run with b^d remaining 0 all the way to the verifier; to this end, P is releasing his nonce N_P to the verifier (i.e., $\text{vis}(P_N_P^d, V) = \text{tt}$). From this table of responses in session d , the adversary will retrieve N_P from this session and pass in all future sessions.

In all of P 's possible initial states, P has not responded to the verifier yet (i.e., $\text{vis}(P_r^{j, ch}, V) = \text{ff}$) and the prover makes his bit b visible to the adversary (i.e., $\text{vis}(P_b^j, Adv) = \text{tt}$).

Verifier's Initial States. The verifier's initial state is as follows:

- the verifier: $V_ch^j = ch^j$, $V_r^{j, ch} = \text{val}^{j, ch}$, $V_rec_r^{j, ch} = \text{null}$, $V_rec_{N_P}^j = \text{null}$,
 $\text{finished}^j, ok^j, V_b^j = 0$, $\bigwedge_{u \in \{V_ch^j \mid 1 \leq j \leq n\}, E \in \{Adv, P\}} \text{vis}(u, E) = \text{ff}$;

for each $j \in \{1, \dots, n\}$, for each $ch^j \in ch_set$, with $\text{val}^{j, ch}$ in enum_1^j .

So, $V_ch^j = ch^j$ equates to V setting the challenges for the j -th execution; each ch^j is chosen randomly in ch_set . The responses precomputed by V (i.e., $V_r^{j, ch}$) are set such that they equate those computed by the prover (i.e., the prover and the verifier pre-share the long-term key x). Also, V has received no DB-response $V_rec_r^{j, ch}$, no nonce $V_rec_{N_P}^{j, ch}$, the execution j has not finished, nor was it checked. And, V (implicitly) sees his variables, does not release the challenges yet ($\text{vis}(u, E) = \text{ff}$), and has computed his local responses V_r^j with values $\text{val}^{j, ch}$.

The values $\text{val}^{j, ch}$ inside the initial states of P and V are produced via a function of j , a constant x , and the set ch_set , with outputs uniformly in r_1 , which emulates the pseudorandom function f_x in the SPADE protocol. This we need not model herein, and we just simply generate the values $\text{val}^{j, ch}$ of the response vectors in P and V . The responses are not invertible without the knowledge of x , and as the SPADE attacker does not know x , Adv can only generate his responses over another domain, r_2 .

Adversary's Initial States. The adversary's initial state is as follows:

- the attacker: $Adv_b^j := 0$, $Adv_r^{j, ch} = \text{val}^{j, ch}$, $\text{vis}(Adv_r^j, V) = \text{ff}$, $\text{vis}(Adv_b^j, V) = \text{ff}$, $\text{vis}(Adv_N_P^j, V) = \text{tt}$
for each $j \in \{1, \dots, n\}$, for each $ch \in ch_set$, with $\text{val}^{j, ch}$ in enum_2^j .

So, Adv has not yet set his bits b^j in corrupted mode (i.e., to 1), has computed his responses $Adv_r^{j, ch}$ (which are over a different domain than the prover's since they do not share the same key x and the same nonce N_P); yet the latter can be overwritten via incoming "help" from a devious prover. Then, Adv 's responses and bit b are not yet visible to V . Yet, Adv can start a full run of the protocol ($b = 0$) of his own accord, i.e., he sends his own nonce N_P to the verifier via $\text{vis}(Adv_N_P^j, V) = \text{tt}$. The latter, together with the initial states of prover, mean that both the attacker and the prover can run the protocol legitimately in any execution j .

Modelling – Part 1: Sending Nonces, b-Setting, Releasing Challenges.

Update commands for Adv – Part 1.

Adv handling the Nonce N_P .

In execution d , if the prover has leaked the whole response-table, then the adversary can retrieve the nonce $P_N_P^d$ and sets it as his next nonce to use.

$$\gamma_1^d : \bigwedge_{ch \in ch_set} \text{vis}(P_r^{d, ch}, Adv) = \text{tt} \rightsquigarrow Adv_N_P^{d+1} := \text{val}(P_N_P^d)$$

In any execution j , if the prover leaks an N_P to the adversary, than the adversary copies it as a nonce of its own for the next execution.

$$\gamma_2^j : \text{val}(P_N_P^j \text{ in } \text{vis}(P_N_P^j, Adv)) = v \rightsquigarrow Adv_N_P^{j+1} := v$$

Then, the adversary forces future executions to run on $b = 1$ and on this learnt nonce (i.e., $Adv_b^{j+1} = 1$, $\text{vis}(Adv_b^{j+1}, V) = \text{tt}$, $\text{vis}(Adv_N_P^{j+1}, V) = \text{tt}$). If the adversary sets this bit to 1, then he also sends an N_P to the

verifier, i.e., $vis(Adv_Np^j, V) = tt$. Recall that for $Adv_b^j = 0$, the adversary can send his Np as part of the initial-state setup.

$$\gamma_3^j : \bigvee_{v \in \text{nonce}_1} (Adv_Np^j = v) \rightsquigarrow Adv_b^j := 1, vis(Adv_b^j, V) := tt, vis(Adv_Np^j, V) := tt$$

Adv handling bit b .

In any execution $j \geq 2$, if the prover sends their bit b as 0, then depending on whether the adversary has learnt a valid Np from P , the adversary will let this bit be 0 (i.e., P can run the protocol-execution entirely), or the adversary will flip it (i.e., Adv will run as P without any timed phase).

$$\begin{aligned} \gamma_4^{bj} : val(P_b^j \text{ in } vis(P_b^j, Adv)) = 0 \wedge (\bigvee_{v \in \text{nonce}_1} (Adv_Np^{j-1} = v)) \rightsquigarrow Adv_b^j := 1, vis(Adv_b^j, V) := tt \\ \gamma_5^{bj} : val(P_b^j \text{ in } vis(P_b^j, Adv)) = 0 \wedge \neg(\bigvee_{v \in \text{nonce}_1} (Adv_Np^{j-1} = v)) \rightsquigarrow Adv_b^j := 0, vis(Adv_b^j, V) := tt \end{aligned}$$

Update commands for P – Part 1. For each honest execution $k \neq d$, the prover sends its nonce to the verifier.

$$\gamma_6^k : P_b^k = 0 \rightsquigarrow vis(P_Np^j, V) := tt$$

Note that the case of P releasing Np to V in session d was treated in the initial-states' setup.

Update commands for V – Part 1. The verifier copies the nonces it receives, be it from P or Adv , in his local duplicate.

$$\begin{aligned} \gamma_7^j : val(Adv_Np^j \text{ in } vis(Adv_Np^j, V)) = v \rightsquigarrow V_recNp^j := v \\ \gamma_8^j : val(P_Np^j \text{ in } vis(P_Np^j, V)) = v \rightsquigarrow V_recNp^j := v \end{aligned}$$

The verifier copies the adversary's sent bit b as his own.

$$\gamma_9^{bj} : val(Adv_b^j \text{ in } vis(Adv_b^j, V)) = b \rightsquigarrow V_b^j := b$$

Then the verifier acts as expected w.r.t. bit b . If the bit b the verifier receives is 0, then V sends challenges.

$$\gamma_{10}^{bj} : V_b^j = 0 \rightsquigarrow \bigwedge_{E \in \{P, Adv\}} vis(V_ch^j, E) := tt$$

Otherwise, for each execution $j \geq 2$, if the bit b the verifier receives is 1, then he looks at the nonce received and compares it with the one in the previous run. If the two coincide, then the adversary succeeds, otherwise he fails.

$$\begin{aligned} \gamma_{11}^{bj} : V_b^j = 1 \wedge V_recNp^{j-1} = V_recNp^j \rightsquigarrow finished^j := tt, ok^j := tt \\ \gamma_{12}^{bj} : V_b^j = 1 \wedge V_recNp^{j-1} \neq V_recNp^j \rightsquigarrow finished^j := tt, ok^j := ff \end{aligned}$$

Modelling – Part 2: Sending the Responses to Challenges.

Update commands for P – Part 2. For each execution j , if $b = 0$, then for any challenge ch received from the verifier specification (which is possible that P is close-by V), the prover answers with his respective responses to the verifier.

$$\gamma_{13}^{ch,j} : P_b^j = 0 \wedge P_far^j = ff \wedge val(V_ch^j \text{ in } vis(V_ch^j, P)) = ch \rightsquigarrow vis(p_r^{ch,j}, V) := tt$$

Update commands for Adv – Part 2. For each execution j , if $b = 0$, then for any challenge ch received from the verifier specification, the adversary answers with his respective responses to the verifier.

$$\gamma_{14}^{ch,j} : Adv_b^j = 0 \wedge val(V_ch^j \text{ in } vis(V_ch^j, Adv)) = ch \rightsquigarrow vis(Adv_r^{ch,j}, V) := tt$$

Note that we do not model the adversary impersonating the verifier (i.e., sending challenges to an afar prover), as this is not part of the threat model we are interested in.

Update *commands for V – Part 2*. The verifier checks the responses as follows. For each value $ch \in ch_set$, there is a command as per the below, which says that the prover is authenticated in the timed phase of execution j .

$$\gamma_{15}^{ch,j} : V_b^j = 0 \wedge V_r^{j,ch} = v \wedge val(P_r^{ch,j} \text{ in } vis(P_r^{ch,j}, V) = v \rightsquigarrow finished^j := tt, ok^j := tt$$

Equally, if the responses sent by the prover are incorrect, then the prover is not authenticated (i.e., $ok^j := ff$).

$$\gamma_{16}^{ch,j} : V_b^j = 0 \wedge V_r^{j,ch} = v \wedge val(P_r^{ch,j} \text{ in } vis(P_r^{ch,j}, V) \neq v \rightsquigarrow finished^j := tt, ok^j := ff$$

The same sets of commands are added for responses coming from the adversary.

$$\gamma_{17}^{ch,j} : V_b^j = 0 \wedge V_r^{j,ch} = v \wedge val(Adv_r^{ch,j} \text{ in } vis(Adv_r^{ch,j}, V) = v \rightsquigarrow finished^j := tt, ok^j := tt$$

$$\gamma_{18}^{ch,j} : V_b^j = 0 \wedge V_r^{j,ch} = v \wedge val(Adv_r^{ch,j} \text{ in } vis(Adv_r^{ch,j}, V) \neq v \rightsquigarrow finished^j := tt, ok^j := ff$$

The TF-encoding Formula. We use atoms:

- $auth_via_help$ as: $(P_far^d \wedge P_collude^d) \rightarrow ok^d$
- $auth_without_help$ as: $\bigwedge_{j \in \{d+1, \dots, n\}} (P_far^j \wedge ok^j)$

The reading of these formulas are as follows. Formula $auth_via_help$ denotes that the attacker authenticates as a far-away prover in one session, with the help of said far-away prover who is corrupted/colluding. Predicate $auth_without_help$ denotes that the attacker authenticates as a far-away prover in one session, without the help of said far-away prover.

Then, our *ATL* formula encoding TF-resistance is: for any strategy the adversary may have, they can never authenticate by themselves by impersonating far-away prover P , and for any collusion between the far-away prover P and attacker Adv such that this collusion eventually removes Adv 's inability to impersonate and some execution d ends successfully authenticating the far-away prover P a, there exist a strategy by Adv alone such that he can pass in a subsequent execution j as a simple man-in-the-middle, without P 's help:

$$[[Adv]]G(\neg auth_without_help) \wedge [[P, Adv]]((\neg auth_without_help \wedge \neg auth_via_help) W (auth_via_help \wedge \langle\langle Adv \rangle\rangle G auth_via_help)) \quad (1)$$

To show the protocol is secure (i.e., TF-resistant), the formula above has to hold.

In *ATL**, the formula to check to find a TF-attack is as follows:

$$\langle\langle P, Adv \rangle\rangle F(auth_via_help \wedge \langle\langle Adv \rangle\rangle G \neg auth_without_help),$$

which means that there exists a collusion between the corrupt prover P and the adversary Adv such that if it eventually leads to the adversary impersonating P then there is a strategy by adv alone such that they always impersonate by themselves.

If this formula holds, then there is a TF-attack, and the protocol is not TF-resistant (i.e., it is not secure).

4.3 Faithful Security Modelling in Our General Semantics

We have a series of notes to empirically ascertain this faithfulness.

- (1) **No Information Forks within Coalitions.** The encodings above are in the specific class of RMI *with visibility control* and subclass of *ATL* formulae introduced in Section 5.2. What is also important to notice is that in the RMI *with visibility control* we built here, there are no information-forks for coalitions $\{P, Adv\}$ and $\{Adv\}$ clearly, as per needed for the decidability result in upcoming Section 5.2.
- (2) **Subjective Interpretations w.r.t. the Attacker.** Moreover, recall that in our work here we considered both the subjective and objective interpretation of *ATL*. In this security examples, the “update” commands we included

in the TF-attacker make sense in the subjective interpretation, so checking our TF-encoding formula on this security example is meaningful to our wider semantics.

- (3) **Imperfect Information.** Our modelling here clearly operates under imperfect information.
- (4) **Perfect Recall Semantics.** The attacker-model requires that we use perfect-recall semantics in deciding the satisfaction of formula (1). By allowing the prover to be either close or far in each of the $n - 1$ sessions in which he does not collude (i.e., by not imposing that the prover participates in each execution), the attacker can interleave executions as they please, produce arbitrary messages, delay their sending to the verifier, etc. Intuitively, this causes paths to not necessarily finish/stutter in one state corresponding to some final protocol-stage. That is, we cannot safely bound the recall, if we wish to capture this attacker-model faithfully. So, w.r.t. formula (1) to check, the attacker agent *Adv* will need to consider his observations “from scratch”, in a perfect recall sense, when executing his strategy for the subgoal *G* cannot_authenticate_without_help.
- (5) On a lesser note, our attacker can interleave executions as he pleases, produce arbitrary messages, delay their sending to the verifier, etc. So, in a practical-verification setting, the formula above, would be checked under certain fairness conditions encoding, e.g., that the verifier is not delayed forever.

5 DECIDABILITY WITHOUT INFORMATION FORKS

In this section we give the proof of Theorem 8, which involves the construction of two turn-based, zero-sum games in which the protagonist (who has imperfect information) captures exactly the strategic abilities of some coalition *A* that has DS-knowledge. The first turn-based game is a version of the information-set construction from [18], which makes it a two-player game structure played between a “protagonist” and an “antagonist” and having an infinite set of states, each state corresponding to a history of the original iCGS and a partial strategy for coalition *A*. The fact that the strategic abilities of the protagonist capture exactly the strategic abilities of coalition *A* in the original game structure is proved here by the use of a variant of alternating bisimulations with imperfect information [7].

The second turn-based game structure is a finite-state abstraction of the first, in which, for each history in an information set, we keep just the first and the last state of each history. This abstraction is not faithful: there is a natural mapping of strategies in the abstraction to strategies in the information set construction such that for any goal which can be enforced by a strategy in the abstraction, the image of that strategy in the information set construction will enforce the same objective, but the opposite does not hold. We show that, when the coalition *A* has DS-knowledge in the original iCGS, the opposite also holds, and the above natural mapping is in fact a bisimulation, hence any objective which is enforced by a strategy in the information set construction can also be enforced in the finite-state abstraction.

This section is organized as follows: we first define a turn-based version of iCGS, which are needed since we utilize results for turn-based games with imperfect information in [25, 51, 53], but our original iCGS are concurrent. We then adapt the alternating bisimulation in [7, 8] in a way which allows simulating the strategic abilities of a coalition *A* in some iCGS by the strategic abilities of a coalition *A'*, not necessarily the same, in another iCGS. We further proceed to the technical construction of the two two-player game arenas described above, for which we build a bisimulation between coalition *A* in the original iCGS and the protagonist in the finite-state two-player game arena. We then prove Theorem 8 for formulas without nesting coalitions, and finally attack the proof of Theorem 8 for the general case.

5.1 Turn-based iCGS

We immediately present the turn-based version of an iCGS, over a coalition *A*.

DEFINITION 13 (TURN-BASED iCGS). Given an iCGS $\mathbf{M} = \langle S, S_0, \{Act_a\}_{a \in Ag}, \pi, \{\sim_a\}_{a \in Ag}, P, \tau \rangle$ and coalition $A \subseteq Ag$, the corresponding (strict alternating) A -turn-based iCGS $TB(\mathbf{M}, A) = \langle S', S_0, \{Act_a \cup \{skip\}\}_{a \in Ag}, \pi', \{\sim'_a\}_{a \in Ag}, P', \tau' \rangle$ is defined as follows:

- $S' = S \cup \{(s, \vec{\alpha}_A) \mid s \in S, \vec{\alpha}_A \in Act_A\}$.
- For every $s' \in S$, $\pi'(s') = \pi(s')$.
- For every $s'_1, s'_2 \in S'$, $a \in Ag$,

$$s'_1 \sim'_a s'_2 \quad \text{iff} \quad s'_1, s'_2 \in S \text{ and } s'_1 \sim_a s'_2, \text{ or} \\ s'_1 = (s_1, \vec{\alpha}_A), s'_2 = (s_2, \vec{\alpha}'_A), \text{ and } s_1 \sim_a s_2.$$

- For every $s' \in S'$ and $a \in Ag$,

$$P'(s', a) = \begin{cases} P(s', a) & \text{if } s' \in S \text{ and } a \in A; \\ P(s, a) & \text{if } s' = (s, \vec{\alpha}_A) \text{ and } a \in \bar{A}; \\ skip & \text{otherwise.} \end{cases}$$

- For every $s' \in S'$, $\vec{\alpha}_A \in Act_A$, and $\vec{\alpha}_{\bar{A}} \in Act_{\bar{A}}$,

$$\tau'(s', \vec{\alpha}_A, \vec{\alpha}_{\bar{A}}) = \begin{cases} (s', \vec{\alpha}_A) & \text{if } s' \in S \text{ and } \vec{\alpha}_{\bar{A}} = skip; \\ \tau(s, \vec{\alpha}'_A, \vec{\alpha}_{\bar{A}}) & \text{if } s' = (s, \vec{\alpha}'_A) \text{ and } \vec{\alpha}_A = skip; \\ \text{undefined} & \text{otherwise.} \end{cases}$$

A turn-based iCGS is *strictly alternating* as, by definition of the transition function τ' , states in $S \subseteq S'$, where it is coalition A 's turn to act, are followed by states of the form $(s, \vec{\alpha}_A)$, where is \bar{A} 's turn, and vice-versa. Indeed, by definition of the protocol function, in states $s \in S \subseteq S'$, agents in coalition \bar{A} can only perform *skip*, whereas in states $(s, \vec{\alpha}_A)$ agents in A can only *skip*. In what follows, for readability, we drop *skip* actions whenever the transition is clear by the context, writing $\tau'(s', \vec{\alpha}_{\bar{A}})$ whenever $s' \in S' \setminus S$ and hence coalition A can only play *skip*.

We then have a bijection between the traces in \mathbf{M} and the traces in $TB(\mathbf{M})$, which maps a trace $tr = s_1 \vec{\alpha}_1 s_2 \vec{\alpha}_2 \dots$ to $tr' = s_1 (\vec{\alpha}_1)_A (s_1, (\vec{\alpha}_1)_{\bar{A}}) (\vec{\alpha}_1)_{\bar{A}} s_2 (\vec{\alpha}_2)_A (s_2, (\vec{\alpha}_2)_{\bar{A}}) (\vec{\alpha}_2)_{\bar{A}} \dots$ where $\vec{\alpha}_i = (\vec{\alpha}_{iA}, \vec{\alpha}_{i\bar{A}})$ is the decomposition of an action profile into the profile of actions of agents in A and the profile of actions of agents in \bar{A} .

Note also that in Def. 13, the labelling function is only defined on states in $S \subseteq S'$. The reason is that we want that an LTL formula ψ be satisfied by a path p in \mathbf{M} if and only if it is satisfied by the corresponding path p' in $TB(\mathbf{M})$. This is because, in a turn-based iCGS, a path p is defined as a (finite or infinite) sequence $s_1 s_2 \dots$ of states in S such that for every $1 \leq i < |p|$ there exist joint actions $\vec{\alpha}_A, \vec{\alpha}_{\bar{A}}$ such that $s'_i = \tau'(s_i, \vec{\alpha}_A) = (s_i, \vec{\alpha}_A)$ and $s_{i+1} = \tau'(s'_i, \vec{\alpha}_{\bar{A}}) = \tau(s_i, \vec{\alpha}_A, \vec{\alpha}_{\bar{A}})$. Then, an A -history (resp. \bar{A} -history) is an history that ends in a state in which is coalition A 's turn (resp. \bar{A} 's turn). We denote the set of all A -histories (resp. \bar{A} -histories) in an iCGS \mathbf{M} as $Hist_A(\mathbf{M})$ (resp. $Hist_{\bar{A}}(\mathbf{M})$).

Given Def. 13 of an A -turn-based iCGS, the satisfaction relation \models for ATL^* -formulas is defined exactly as in Def. 6, by considering the suitable notion of path.

5.2 Bisimulation

In this section we adapt the notion of bisimulation between two iCGS [7, 8] in such a way that strategic abilities of a coalition A in a system may be simulated by strategic capabilities of a perhaps different coalition A' in another system.

This notion is also modified to accommodate turn-based iCGS. We start by adapting the notion of partial strategies from [8] for turn-based iCGS.

Partial strategies in turn-based iCGS. A *partial (uniform, memoryful) strategy* for an agent $a \in Ag$ in a coalition $A \subseteq Ag$ is a partial function $\sigma_a : Hist_A(M) \rightarrow Act_a$ such that for each $h, h' \in Hist_A(M)$:

- (1) Whenever $\sigma_a(h)$ is defined, then $\sigma_a(h) \in P(lst(h), a)$.
- (2) $h \sim_a h'$ implies $\sigma_a(h) = \sigma_a(h')$.

We denote the domain of partial strategy σ_a as $dom(\sigma_a)$.

Given a coalition $A \subseteq Ag$, a *partial strategy for A* is a tuple σ_A of partial strategies, one for each agent $a \in A$. The set of partial strategies for A is denoted as $PStr_A$. Given a coalition $A \subseteq Ag$ and a set $H \subseteq Hist_A(M)$ of A -histories, we denote by $PStr_A(H)$ the set of partial strategies whose domain is H :

$$PStr_A(H) = \{\sigma_A \in PStr_A \mid dom(\sigma_a) = H \text{ for all } a \in A\}$$

Additionally, given a (total or partial) strategy σ_A , a history $h \in dom(\sigma_A)$ and an action profile $\vec{\alpha} \in P(\tau(lst(h), \sigma_A(h)), \bar{A})$, we consider the following successor functions:

$$\begin{aligned} \text{hext}(h, \sigma_A) &= h \cdot \tau(lst(h), \sigma_A(h)) \\ \text{hext}(h, \sigma_A, \vec{\alpha}) &= h \cdot \tau(\tau(lst(h), \sigma_A(h)), \vec{\alpha}) \\ \text{hext}^2(h, \sigma_A) &= \{\text{hext}(h, \sigma_A, \vec{\alpha}) \mid \vec{\alpha} \in Act_{\bar{A}}\} \end{aligned}$$

$\text{hext}(h, \sigma_A)$ stands for the one-step extension, in the turn-based game structure, of history h with the action provided by strategy σ_A , before the agents in \bar{A} move, while $\text{hext}(h, \sigma_A, \vec{\alpha})$ is the extension of $\text{hext}(h, \sigma_A)$ by the move $\vec{\alpha}$ played by agents in \bar{A} . Also $\text{hext}^2(h, \sigma_A)$ stands for the set of two-step extensions of h by strategy σ_A and all moves $\vec{\alpha}$ played by agents in \bar{A} .

When $h \in Hist_A(M)$ ends in an \bar{A} state and $\alpha \in P(lst(h), \bar{A})$, we also denote

$$\text{hext}(h, \vec{\alpha}) = h \cdot \tau(lst(h), \vec{\alpha})$$

A useful identity which will be used in the sequel is the following:

$$lst(\text{hext}(h, \sigma, \beta)) = \tau(lst(h), \sigma, \beta) \quad (2)$$

We also define n -step, finite and infinite outcomes of a strategy σ_A as

$$\begin{aligned} out^n(\sigma_A) &= \bigcup \{\text{hext}^2(h, \sigma_A) \mid h \in dom(\sigma_A), |h| = n\} & out^{\leq n}(\sigma_A) &= \bigcup_{k \leq n} out^k(\sigma_A) \\ out(\sigma_A) &= \{\lambda \in Paths(M) \mid \forall n \in \mathbb{N}, \lambda_{\leq n} \in out^n(\sigma_A)\} & out^f(\sigma_A) &= \bigcup_{n \in \mathbb{N}} out^n(\sigma_A) \\ out^f(h, \sigma_A) &= \{k \in out^f(\sigma_A) \mid h \leq k\} & out(h, \sigma_A) &= \{\lambda \in out(\sigma_A) \mid h \leq \lambda\} \end{aligned}$$

Hereafter $M = \langle S, S_0, \{Act_a\}_{a \in Ag}, \pi, \{\sim_a\}_{a \in Ag}, P, \tau \rangle$ and $M' = \langle S', S'_0, \{Act'_a\}_{a \in Ag}, \pi', \{\sim'_a\}_{a \in Ag}, P', \tau' \rangle$ are two turn-based iCGS for $A \subseteq Ag$ and $A' \subseteq Ag'$, respectively. In the sequel, the notations hext' refer to successor functions on histories of M' .

The following definition, introduced in [7, 8], models the simulation of “local” strategies of the coalition A in M with “local” strategies of A' in M' , where the locality in both iCGS refers to some equivalence class for common knowledge. The idea is that, assuming some relationship exists between two common knowledge neighbourhoods $C_A(h)$ in M and,

resp., $C_{A'}(h')$ in \mathcal{M}' , we would like to extend that relationship to a simulation of each strategy defined on each history in $C_A(h)$ by a strategy defined on each history in $C_{A'}(h')$:

DEFINITION 14 (STRATEGY SIMULATORS). *An (A, A') -strategy simulator (or simply strategy simulator, when A, A' are understood from the context) is a family $ST = (ST_{C_A(h), C_{A'}(h')})_{h \in \text{Hist}_A(\mathcal{M}), h' \in \text{Hist}_{A'}(\mathcal{M}')} of mappings $ST_{C_A(h), C_{A'}(h')} : PStr_A(C_A(h)) \rightarrow PStr_{A'}(C_{A'}(h'))$ such that for all histories $h, k \in \text{Hist}(\mathcal{M})$ and $h', k' \in \text{Hist}(\mathcal{M}')$,$*

$$\text{if } C_A(h) = C_A(k) \text{ and } C_{A'}(h') = C_{A'}(k'), \text{ then } ST_{C_A(h), C_{A'}(h')} = ST_{C_A(k), C_{A'}(k')} \quad (3)$$

Hereafter, we simplify the notation by writing $ST(\sigma)$ instead of the cumbersome $ST_{C_A(h), C_{A'}(h')}(\sigma)$, whenever h and h' are clear from the context and $\sigma \in PStr(C_A(h))$.

We can now adapt the history-based version of (bi)simulation for turn-based iCGS from [8]. A simulation is, as usual, a binary relation between a history in \mathcal{M} and a history in \mathcal{M}' , which has to be compatible with strategy simulators and with reactions by the respective anti-coalition – reactions which are modeled by "reverse simulation maps":

DEFINITION 15 (MEMORYFUL SIMULATION). *Let $A \subseteq Ag$, $A' \subseteq Ag'$ be coalitions of agents. A relation $\Rightarrow_{A, A'} \subseteq (\text{Hist}_A(\mathcal{M}) \times \text{Hist}_{A'}(\mathcal{M}')) \cup (\text{Hist}_{\bar{A}}(\mathcal{M}) \times \text{Hist}_{\bar{A}'}(\mathcal{M}'))$ is a simulation for A, A' (denoted \Rightarrow in the sequel) iff there exists*

- a. a strategy simulator ST ;
- b. a family of reverse simulation maps $(\rho_{h, h'})_{(h, h') \in \mathcal{J}}$ with $\mathcal{J} = \{(h, h') \in \text{Hist}_{\bar{A}}(\mathcal{M}) \times \text{Hist}_{\bar{A}'}(\mathcal{M}') \mid h \Rightarrow h'\}$ and $\rho_{h, h'} : \text{Act}_{\bar{A}'} \longrightarrow \text{Act}_{\bar{A}}$;

such that, for any two histories $h \in \text{Hist}(\mathcal{M})$, $h' \in \text{Hist}(\mathcal{M}')$, $h \Rightarrow h'$ implies the following properties:

- (Sim.1) $\pi(\text{lst}(h)) = \pi'(\text{lst}(h'))$;
- (Sim.2) If $h \in \text{Hist}_A(\mathcal{M})$ and $h' \in \text{Hist}_A(\mathcal{M}')$ then for each $\sigma_A \in PStr_A(C_A(h))$, $\text{hext}(h, \sigma_A(h)) \Rightarrow \text{hext}'(h', ST(\sigma_A)(h'))$.
- (Sim.3) If $h \in \text{Hist}_{\bar{A}}(\mathcal{M})$ and $h' \in \text{Hist}_{\bar{A}'}(\mathcal{M}')$ then for each $\alpha' \in P(\bar{A}', \text{lst}(h'))$, $\text{hext}(h, \rho_{h, h'}(\alpha')) \Rightarrow \text{hext}'(h', \alpha')$.

Additionally:

- (Sim.4) For each $h \in \text{Hist}_{\bar{A}}(\mathcal{M})$, $h' \in \text{Hist}_{\bar{A}'}(\mathcal{M}')$, $k \in \text{Hist}_{\bar{A}}(\mathcal{M})$ and $k' \in \text{Hist}_{\bar{A}'}(\mathcal{M}')$ with $h \Rightarrow h'$, $k \Rightarrow k'$, $h \sim_A^C k$, and $k' \sim_{A'}^C h'$, for each $\alpha' \in P(\text{lst}(h'), \bar{A}')$ and $\beta' \in P(\text{lst}(k'), \bar{A}')$,

$$\text{hext}'(h', \alpha') \sim_{A'}^C \text{hext}'(k', \beta') \text{ implies } \text{hext}(h, \rho_{h, h'}(\alpha')) \sim_A^C \text{hext}(k, \rho_{k, k'}(\beta'))$$

A relation $\Leftrightarrow_{A, A'}$ is a bisimulation iff both $\Rightarrow_{A, A'}$ and its converse $\Leftrightarrow_{A, A'}^{-1} = \{(h', h) \mid h \Leftrightarrow_{A, A'} h'\}$ are simulations.

We also extend (bi)-simulation to paths $\lambda \in \text{Path}(\mathcal{M})$, $\lambda' \in \text{Path}(\mathcal{M}')$, by denoting $\lambda \Rightarrow \lambda'$ iff for all $j \geq 0$, $\lambda_{\leq j} \Rightarrow \lambda'_{\leq j}$.

The following lemma adapts Lemma 9 in [8]. To do so, for a set $\chi \subseteq X \times Y$ of tuples, we denote the projection on the first component as $\chi|_1 = \{x \in X \mid \exists y \in Y \text{ with } (x, y) \in \chi\}$, and we employ a similar notation for the projection on the second component $\chi|_2$. Also we employ the term *common knowledge class* for A to designate some set of histories H which is the common knowledge neighborhood $H = C_A(h)$ of some history h .

LEMMA 3. *Suppose that \Rightarrow is a simulation for A, A' . Consider two common knowledge classes $H_0 \in \text{Hist}_A(\mathcal{M})$ and $H'_0 \in \text{Hist}_{A'}(\mathcal{M}')$ and suppose there exists a set of pairs of histories $\chi \subseteq C_A(h_0) \times C_{A'}(h'_0)$ and some mapping $\text{rev}_\chi : \chi|_2 \longrightarrow \chi|_1$ with $\text{rev}_\chi(h') \Rightarrow h'$ for each $h' \in H'_0$.*

Then for every strategy σ_A , there exists a strategy $\sigma'_{A'}$ such that, for any $(h, h') \in \chi$,

() for every path $\lambda' \in \text{out}(h', \sigma'_{A'})$ there exists a path $\lambda \in \text{out}(h, \sigma_A)$ such that $\lambda \Rightarrow \lambda'$.*

The proof for this is an adaptation of the proofs in [7, 8], and can be found in Appendix A.

We may now state and prove the main result of this section:

THEOREM 11. *In the context of the notations of this section, let $h \in \text{Hist}(\mathbb{M})$ and $h' \in \text{Hist}(\mathbb{M}')$ be histories and $\Rightarrow_{A,A'}$ a simulation for A, A' such that $h \Rightarrow_{A,A'} h'$ and for any $h'_1 \in \text{Hist}(\mathbb{M})$ with $h'_1 \sim_{A'}^E h$, there exists $h_1 \in \text{Hist}(\mathbb{M})$ with $h_1 \sim_A^E h$ and $h_1 \Rightarrow_{A,A'} h'_1$.*

Then for any path formula ψ , if $(\mathbb{M}, \text{lst}(h)) \models \langle\langle A \rangle\rangle\psi$ then $(\mathbb{M}', \text{lst}(h')) \models \langle\langle A' \rangle\rangle\psi$.

PROOF. We give the proof for the subjective semantics of ATL as it will immediately entail the result for the objective semantics.

Since $(\mathbb{M}, \text{lst}(h)) \models \langle\langle A \rangle\rangle\psi$, we may find a strategy σ_A such that, for any $h_1 \sim_A^E h$ and $\lambda \in \text{out}(h_1, \sigma_A)$, $(\mathbb{M}, \lambda_{\geq |h|}) \models \psi$. Define

$$\chi = \{(h_1, h'_1) \mid h_1 \sim_A^E h, h'_1 \sim_{A'}^E h', h_1 \Rightarrow_{A,A'} h'_1\}$$

We may then apply Lemma 3 and conclude that there exists $\sigma_{A'}$ such that, for any $h'_1 \sim_{A'}^E h'$ and any path $\lambda' \in \text{out}(h'_1, \sigma_{A'})$, there exists $h_1 \sim_A^E h$ and $\lambda \in \text{out}(h_1, \sigma_A)$ such that $\lambda \Rightarrow_{A,A'} \lambda'$.

Now observe that $\lambda \Rightarrow_{A,A'} \lambda'$ implies that $\pi(\lambda[i]) = \pi(\lambda'[i])$ for all $i \in \mathbb{N}$. Hence the projection of both paths onto AP is identical, $\lambda|_{AP} = \lambda'|_{AP}$, which implies that they satisfy the same LTL formulas using atoms in AP . By an easy induction on the number of nested coalition operators in the ATL formula φ , we also get that $\lambda \models \psi$ if and only if $\lambda' \models \psi$. Hence $\sigma_{A'}$ is a witness that $(\mathbb{M}', \text{lst}(h')) \models \langle\langle A' \rangle\rangle\psi$. \square

Note that the above result also holds for the objective semantics [43], since in that case we only need to consider the singleton set $\chi = \{(h, h')\}$.

5.3 Model-checking formulas with non-nesting coalition operators

In this subsection we give the two main constructions of turn-based games that simulate strategic abilities of a given coalition A with DS-knowledge. We start with the adaptation of the information set construction from [18].

Assume that $\mathbb{M} = \langle Ag, S_A, S_{\bar{A}}, S_0, Act, \{\sim_a\}_{a \in Ag}, P, \tau, \pi \rangle$ is a (turn-based) iCGS, and suppose that S_0 is closed under $\sim_{A'}^C$, that is, if $s \sim_{A'}^C s'$ and $s \in S_0$ then $s' \in S_0$. A useful notation that we need in the sequel is the set of partial strategies whose domain is the union of all common knowledge neighborhoods of some prefixes of a history h and which are compatible with some given (full) strategy σ :

$$PStr_A(\sigma, \leq h) = \bigcup_{h' \leq h} PStr_A(C_A(h') \cap \text{out}^f(\sigma, Q_0))$$

We build a two-player (infinite state) turn-based game structure $\hat{\mathbb{M}}$ in which the protagonist is denoted 1 and the antagonist is 2, such that the turn-based iCGS $TB(\mathbb{M}, A)$ and $\hat{\mathbb{M}}$ are $A, 1$ -bisimilar.

The two-player game structure is defined as $\hat{\mathbb{M}} = (\hat{Q}_1, \hat{Q}_2, S_0, \sim_1, Act_1, Act_2, \hat{\tau}, \hat{P}, \hat{\pi})$, where :

- The set of states is:

$$\begin{aligned} \hat{Q}_1 &= S_0 \cup \{[\sigma, h, \beta] \mid \text{there exists } \sigma_0 \in Str_A(\mathbb{M}) \text{ with } \sigma \in PStr_A(\sigma_0, \leq h) \text{ and } \beta \in P(\bar{A}, h)\} \\ \hat{Q}_2 &= \{[\sigma, h] \mid \text{there exists } \sigma_0 \in Str_A(\mathbb{M}) \text{ with } \sigma \in PStr_A(\sigma_0, \leq h)\} \end{aligned}$$

Here, a macro-state $[\sigma, h, \beta]$ encodes the information set $C_A(\text{hext}(h, \sigma, \beta))$. Also a macro-state $[\sigma, h]$ encodes the information set $C_A(h) \cap \text{out}^{|h|}(\sigma, Q_0)$. The σ component in these macro-states is called the *strategy commitment for A*.

- The set of actions are $Act_1 = \bigcup_{n \in \mathbb{N}} Act_1^n$ where $Act_1^n = \bigcup_{k \in \text{Hist}^n(\mathbb{M})} PStr_A(C_A(k))$, and $Act_2 = Act_{\bar{A}}$.
- The indistinguishability relation \sim_1 is defined as:
 - $[\sigma_1, h_1] \sim_1 [\sigma_2, h_2]$ iff $\sigma_1 = \sigma_2$ and $h_1 \sim_A^C h_2$.
 - $[\sigma_1, h_1, \beta_1] \sim_1 [\sigma_2, h_2, \beta_2]$ iff $\sigma_1 = \sigma_2$, and $\text{hext}(h_1, \sigma_1, \beta_1) \sim_A^C \text{hext}(h_2, \sigma_2, \beta_2)$.
 - For each $s_1, s_2 \in S_0$, $s_1 \sim_1 s_2$ iff $s_1 \sim_A^C s_2$.

Note that, for two macro-states to be indistinguishable for the protagonist, they must be produced by the same strategy commitment, which allows for this indistinguishability to be an equivalence relation. Also note that the second type of indistinguishability implies that $h_1 \sim_A^C h_2$, since \sim_A^C is a perfect recall and synchronous common knowledge indistinguishability relation.

- The protocol and the transition function are given as:
 - For each $s \in S_0$, $\hat{P}(1, h) = PStr_A(C_A(s))$.
 - For each $\sigma \in \hat{P}(1, h) = PStr_A(C_A(s))$, $\hat{\tau}(s, \sigma) = [\sigma, s]$.
 - For each $[\sigma, h, \beta] \in \hat{Q}_1$, $\hat{P}(1, [\sigma, h, \beta]) = PStr_A(C_A(\text{hext}(h, \sigma, \beta)))$.
 - For each $\sigma_1 \in \hat{P}(1, [\sigma, h, \beta]) = PStr_A(C_A(\text{hext}(h, \sigma, \beta)))$,

$$\hat{\tau}([\sigma, h, \beta], \sigma_1) = [\sigma \otimes \sigma_1, \text{hext}(h, \sigma, \beta)]$$

where $\sigma \otimes \sigma_1$ is the one-step extension of the strategy commitment σ by σ_1 defined as follows: $\text{dom}(\sigma \otimes \sigma_1) = \text{dom}(\sigma) \cup \text{dom}(\sigma_1)$ and

$$(\sigma \otimes \sigma_1)(h) = \begin{cases} \sigma(h) & \text{for } h \in \text{dom}(\sigma) \\ \sigma_1(h) & \text{for } h \in \text{dom}(\sigma_1) \end{cases}$$

Informally, in each macro-state $[\sigma, h, \beta]$, the protagonist must propose an A -profile of actions to be played at $\text{hext}(h, \sigma, \beta)$, and also action profiles which can be "consistently" played on each history $h' \sim_A^C \text{hext}(h, \sigma, \beta)$. The role of the partial strategy σ_1 is to gather together all these action profiles, by which the current strategy commitment σ is extended.

Note additionally that the set of actions for the protagonist is infinite, but, in each macro-state of $\hat{\mathbb{M}}$, only finitely many actions are available as per definition of the protocol mapping \hat{P} .

- For each $[\sigma, h] \in \hat{Q}_2$, $\hat{P}(2, [\sigma, h]) = P(\bar{A}, \text{lst}(h))$, and for each $\beta \in Act_{\bar{A}}$,

$$\hat{\tau}([\sigma, h], \beta) = [\sigma, h, \beta]$$

- The labeling with atomic propositions is:

$$\begin{aligned} \hat{\pi}(s) &= \pi(s) & \text{for } s \in S_0 \\ \hat{\pi}([\sigma, h, \beta]) &= \pi(\tau(\text{lst}(h), \sigma(h), \beta)) & \text{for } [\sigma, h, \beta] \in \hat{Q}_1 \setminus S_0 \end{aligned}$$

Note that, similarly with the turn-based version of an iCGS, we only label states belonging to the protagonist.

REMARK 3. Note that, given a state $[\sigma, h] \in \hat{Q}_2$, there exists a unique history $\hat{h} \in \text{Hist}(\hat{\mathbb{M}})$ with $\text{lst}(\hat{h}) = [\sigma, h]$. A similar property holds for states $[\sigma, h, \beta] \in \hat{Q}_1$, for which there exists a unique history $h_{[\sigma, h, \beta]} \in \text{Hist}_{\mathbb{M}}$ with $|h_{[\sigma, h, \beta]}| = |h| + 1 = n + 1$, $h_{[\sigma, h, \beta]}[\leq n] = h$ and $h_{[\sigma, h, \beta]}[n + 1] = \text{hext}(h, \sigma, \beta)$.

Therefore, in the sequel we will identify histories in $\hat{\mathbb{M}}$ with their final macro-states, be they in \hat{Q}_1 or \hat{Q}_2 .

The above intuitive presentation of each state $[\sigma, h, \beta]$ as being a replica of $\text{hext}(h, \sigma, \beta)$ can be given a very concrete interpretation in terms of bisimulations. Formally, consider the following relation:

- (1) For all $s \in S_0$, $s \iff_{A,1} s$.
- (2) For all $[\sigma, h, \beta] \in \hat{Q}_1$, $\text{hext}(h, \sigma, \beta) \iff_{A,1} [\sigma, h, \beta]$.
- (3) For all $[\sigma, h] \in \hat{Q}_2$, $\text{hext}(h, \sigma) \iff_{A,1} [\sigma, h]$.

THEOREM 12. $\iff_{A,1}$ is a bisimulation for $A, 1$ between $TB(\mathbb{M}, A)$ and $\hat{\mathbb{M}}$.

PROOF. We need to define the strategy simulators and the reverse simulation maps.

- (1) For each pair $\text{hext}(h, \sigma, \beta) \iff_{A,1} [\sigma, h, \beta]$, note first that $P\text{Str}_A(C_A(\text{hext}(h, \sigma, \beta))) = P\text{Str}_1(C_1([\sigma, h, \beta]))$. Then we set the identity map as the strategy simulator, that is, $ST = ST_{C_A(\text{hext}(h, \sigma, \beta)), C_1([\sigma, h, \beta])}(\sigma_1) = \sigma_1$.
- (2) Dually, the opposite strategy simulator $ST^{-1} = ST_{C_1([\sigma, h, \beta]), C_A(\text{hext}(h, \sigma, \beta))}$ is also the identity mapping.
- (3) For each simulation pair $\text{hext}(h, \sigma) \iff_{A,1} [\sigma, h]$, recall that, by definition, $\hat{P}(2, [\sigma, h]) = P(\bar{A}, \text{lst}(h))$. Then we also set the reverse simulation map as the identity map on $P(\bar{A}, \text{lst}(h))$, that is, $\rho_{\text{hext}(h, \sigma), [\sigma, h]}(\beta) = \beta$.
- (4) Dually the reverse simulation map for the opposite simulation is the identity too.

The four properties of bisimulations can be easily checked. For example, for proving property 4 for (the direct and reverse part of) $\iff_{A,1}$, assume we have $[\sigma_1, h_1] \sim_1 [\sigma_2, h_2]$ with $\text{hext}(h_i, \sigma_i) \iff_{A,1} [\sigma_i, h_i]$ for $i = 1, 2$. Then, for any $\beta_1, \beta_2 \in \text{Act}_{\bar{A}}$, $\hat{\tau}([\sigma_1, h_1], \beta_1) \sim_1 \hat{\tau}([\sigma_2, h_2], \beta_2)$ is equivalent with $\text{hext}(h_1, \sigma_1, \beta_1) \sim_A^C \text{hext}(h_2, \sigma_2, \beta_2)$ by construction of $\hat{\tau}$. \square

Now, we give the finite game structure which abstracts $\hat{\mathbb{M}}$ when coalition A has DS-knowledge in the original iCGS. The rough idea is to compact each $\hat{\mathbb{M}}$ -state $[\sigma, h, \beta]$ into a configuration $D \subseteq S^2$ in which we keep only the initial and the final state of each history $h' \in C_A(\text{hext}(h, \sigma, \beta)) \cap \text{out}(h[1], \sigma)$ (the "frontier" of $C_A(\text{hext}(h, \sigma, \beta)) \cap \text{out}(h[1], \sigma)$). Then protagonist actions would correspond with one-step partial uniform strategies for A whose domain is some set of pairs of states D which form a common-knowledge class for A and represent such a "frontier". This would produce a finite game structure which, in general, may lose some information. More specifically, in the absence of the DS-knowledge property, if the protagonist wins this finite game structure for some objective then it also wins $\hat{\mathbb{M}}$, but not the other way round. The reason is that some A -strategy σ in \mathbb{M} may behave differently on two histories h_1, h_2 with $h_1[1] = h_2[1]$, $\text{lst}(h_1) = \text{lst}(h_2)$ and $h_1 \sim_A^C h_2$ (i.e. $\sigma(h_1) \neq \sigma(h_2)$), and yet when A chooses $\sigma(h_2)$ at h_1 she would not produce a winning play – and neither the other way round. This is where DS-knowledge helps: any A -strategy must produce the same action on any two histories $h_1 \sim_A^C h_2$ of equal length and which share the same initial and final state. This will allow us to simulate any A -strategy σ in the original game structure \mathbb{M} with a protagonist strategy in the finite game structure we present below.

The finite two-player game is denoted $\tilde{\mathbb{M}} = (\tilde{Q}_1, \tilde{Q}_2, \tilde{Q}_0 \sim_1, \text{Act}_1, \text{Act}_2, \tilde{\tau}, \tilde{P}, \tilde{\pi})$, and built as follows:

- The set of antagonist states is:

$$\tilde{Q}_2 = \{(D, \sigma, (s_1, s_2)) \mid (s_1, s_2) \in S \times S, D \subseteq C_A(s_1 s_2), \sigma \in P\text{Str}_A(C_A(s_1 s_2))\}$$

Intuitively, D is the set of pairs consisting of the first and the last state of some history in some of the sets of the form $C_A(h) \cap \text{out}(h[1], \sigma')$, where $[\sigma', h] \in \hat{Q}_2$ will be detailed below. In the above definition when we employ $C_A(s_1 s_2)$, $s_1 s_2$ represents a two-state history in \mathbb{M} .

- The set of protagonist states is:

$$\tilde{Q}_1 = S_0 \cup \{(D, \sigma, (s_1, s_2), \beta) \mid (s_1, s_2) \in S \times S, D \subseteq C_A(s_1 s_2), \sigma \in PStr_A(C_A(s_1 s_2)), \beta \in P(\bar{A}, s_2)\}$$

These second type of states simulate macro-states of the form $[\sigma', h, \beta]$ in \hat{S}_1 , in a sense to be detailed below.

- The set of initial states is S_0 .
- For each $(D, \sigma, (s_1, s_2)), (D', \sigma', (s'_1, s'_2)) \in \tilde{Q}_2$, $(D, \sigma, (s_1, s_2)) \sim_1 (D', \sigma', (s'_1, s'_2))$ iff $D = D'$, $\sigma = \sigma'$ and $s_1 s_2 \sim_A^C s'_1 s'_2$.
- For each $(D, \sigma, (s_1, s_2), \beta), (D', \sigma', (s'_1, s'_2), \beta') \in \tilde{Q}_1$, $(D, \sigma, (s_1, s_2), \beta) \sim_1 (D', \sigma', (s'_1, s'_2), \beta')$ iff $D = D'$, $\sigma = \sigma'$ and $\text{hext}(s_1 s_2, \sigma, \beta) \sim_A^C \text{hext}(s'_1 s'_2, \sigma', \beta')$.
- For each $s \in S_0$, $s \sim_1 s'$ iff $s' \in S_0$ and $s \sim_A^C s'$.
- $\tilde{P}(2, (D, \sigma, (s_1, s_2))) = P(\bar{A}, s_2)$ and:

$$\tilde{\tau}((D, \sigma, (s_1, s_2)), \beta) = (D, \sigma, (s_1, s_2), \beta) \text{ for each } \beta \in P(\bar{A}, s_2)$$

- If we denote $s_3 = \tau(s_2, (\sigma(s_1 s_2), \beta))$ then $\tilde{P}(1, (D, \sigma, (s_1, s_2), \beta)) = PStr_A(C_A(s_1 s_3))$ and:

$$\tilde{\tau}((D, \sigma, (s_1, s_2), \beta), \sigma_1) = (D', \sigma_1, (s_1, s_3)) \text{ for each } \sigma_1 \in PStr_A(C_A(s_1 s_3)),$$

where $D' = \{(s_4, s_6) \mid \text{there exists } (s_4, s_5) \in D \text{ with } s_4 s_5 s_6 \in \text{hext}^2(s_4 s_5, \sigma) \text{ and } s_6 \sim_A^C s_3\}$.

- For each $s \in S_0$, $P(s, 1) = PStr_A(C_A(s))$ and for each $\sigma \in PStr_A(C_A(s))$, $\tilde{\tau}(s, \sigma) = (C_A(s), \sigma, (s, s))$.
- The labeling with atoms is:
 - $\tilde{\pi}(s) = \pi(s)$ for any $s \in S_0$.
 - $\tilde{\pi}(D, \sigma, (s_1, s_2), \beta) = \pi(\tau(s_2, (\sigma(s_1 s_2), \beta)))$ for any $(D, \sigma(s_1, s_2), \beta) \in \tilde{Q}_2$.

Note that, in accordance with the notion of turn-based CGS, we only label protagonist states.

Again, we have employed the term "simulation" in the above informal presentation of this construction. Actually, we may define a **bisimulation** between $\hat{\mathbb{M}}$ and $\tilde{\mathbb{M}}$ under the assumption that \mathbb{M} satisfies DS-knowledge. To this end, given a history $h \in \text{Hist}(\mathbb{M})$ and a partial strategy σ with $h \in \text{out}^f(\sigma)$, we define the following two objects:

$$D_{\sigma, h} = \{(s_1, s_2) \in S \times S \mid \text{for some } h' \in C_A(h) \cap \text{out}^f(\sigma), h'[1] = s_1, \text{lst}(h') = s_2\}$$

$$\tilde{\sigma}(s_1 s_2) = \sigma(h') \text{ for each } s_1 s_2 \in C_A(h[1] \cdot \text{lst}(h)) \text{ and } \underline{\text{some}} h' \in C_A(h) \cap \text{out}^f(\sigma) \text{ with } h'[1] = s_1, \text{lst}(h') = s_2$$

REMARK 4. The above definition of $\tilde{\sigma}$ does not depend on the choice of h' when A has DS-knowledge.

To see this, assume we have $h_1, h_2 \in C_A(h) \cap \text{out}(\sigma)$ with $h_1[1] = h_2[1]$, $\text{lst}(h_1) = \text{lst}(h_2)$. Since A has DS-knowledge we must have $h_1 \sim_A^D h_2$. But σ is a uniform strategy for A , hence $\sigma(h_1) = \sigma(h_2)$.

Then the desired bisimulation between $\hat{\mathbb{M}}$ and $\tilde{\mathbb{M}}$ is the following relation:

- (1) For all $[\sigma, h] \in \hat{Q}_2$, $[\sigma, h] \iff_{1,1} (D_{\sigma, h}, \tilde{\sigma}, (h[1], \text{lst}(h)))$.
- (2) For all $[\sigma, h, \beta] \in \hat{Q}_1$, $[\sigma, h, \beta] \iff_{1,1} (D_{\sigma, h}, \tilde{\sigma}, (h[1], \text{lst}(h)), \beta)$.

This formalizes the intuition that $(D_{\sigma, h}, \tilde{\sigma}, (h[1], \text{lst}(h)), \beta)$ contains exactly the tuples of initial and final states of histories h' which have the same length as h , are in the outcome of σ , and are in the same common-knowledge class for A .

Note that this is a relation on pairs of states from $\hat{S} \times \tilde{S}$, but can be straightforwardly extended to a relation on $Hist(\hat{M}) \times Hist(\tilde{S})$ by simply defining $\hat{h} \hat{\equiv}_{1,1} \tilde{h}$ whenever $\hat{h} \hat{\equiv}_{1,1} l_{st}(\tilde{h})$, and recalling that a state $[h, \sigma, \beta]$ uniquely identifies a history in \hat{M} . Hence Theorem 11 can be applied here too.

THEOREM 13. *If coalition A has DS-knowledge in M , then $\hat{\equiv}_{1,1}$ is a bisimulation (for 1, 1) between \hat{M} and \tilde{M} .*

The proof of this theorem, which requires tedious checking of the properties of bisimulations, can be found in the appendix.

We are now in position to define an algorithm for model-checking formulas of the type $\langle\langle A \rangle\rangle\psi$ where ψ is a purely LTL formula and coalition A in the given iCGS has DS-knowledge.

THEOREM 14. *Given an iCGS M of size n which has DS-knowledge, a formula $\langle\langle A \rangle\rangle\psi$ containing no coalition operator and of size m , and a state $s \in S$, there exists a parity game \mathcal{G} of size $O(2^{2^{m^2+n}})$ such that the protagonist has a winning strategy in \mathcal{G} if and only if $(M, s) \models \langle\langle A \rangle\rangle\psi$.*

PROOF. Again we give the proof for the subjective semantics and comment where it must be adapted to handle the objective semantics – which is, in fact, just in the next paragraph.

Denote $Q_0^s = \{s' \in S \mid s' \sim_a s \text{ for some } a \in A\}$ and replace, in M , the set of initial states with Q_0^s . For the case of the objective semantics we simply start with $Q_0^s = \{s\}$. Build the two-player turn-based game structure $\tilde{M}_s = (\tilde{Q}_1, \tilde{Q}_2, Q_0^s, \sim_1, Act_1, Act_2, \tilde{\tau}, \tilde{P}, \tilde{\pi})$. Note that the size of \tilde{M}_s is $O(2^{|S|^2} \cdot |Act|^{|S|^2 \cdot |Ag|} \cdot |S|^2 \cdot |Act|^{|Ag|}) = O(2^{|S|^2 \cdot |Ag| \cdot \log |Act|})$, where S is the set of states in M , and the number of observability classes is also $O(2^{|S|^2 \cdot |Ag| \cdot \log |Act|})$.

Using results from [52], we build a nondeterministic Büchi automaton $\mathcal{A}_\psi = (Q_\psi, q_0^\psi, \delta_\psi, \Pi_\psi, B_\psi)$, with $\delta_\psi : Q_\psi \times 2^{\Pi_\psi} \rightarrow Q_\psi$ and $B_\psi \subseteq Q_\psi$ representing the set of Büchi states. The language of \mathcal{A}_ψ contains exactly the paths in $(\Pi_\psi)^\omega$ which satisfy ψ , that is, $L(\mathcal{A}_\psi) = \{\lambda \in (\Pi_\psi)^\omega \mid \lambda \models \psi\}$. Note that the size of \mathcal{A}_ψ is $2^{|\psi|}$.

We then build a turn-based Büchi game with imperfect information $M_{A,\psi}$, as a (slightly adapted) synchronous product of \tilde{M}_s and \mathcal{A}_ψ . This construction ensures that $(M, s) \models \langle\langle A \rangle\rangle\psi$ iff the protagonist wins $M_{A,\psi}$. The adaptation of the synchronous product is classical, and must account for the fact that \mathcal{A}_ψ is transition-labeled, whereas \tilde{M}_s is state-labeled. Formally, $M_{A,\psi} = (\delta_\psi \times \tilde{Q}_1, \{r_0\} \cup (\delta_\psi \times \tilde{Q}_2), r_0, \bar{\delta}, \Pi_\psi, \bar{B}, \bar{\sigma})$ where:

- (1) Protagonist states are $\delta_\psi \times \tilde{Q}_1$ and antagonist states are $\{r_0\} \cup (\delta_\psi \times \tilde{Q}_2)$. Note that the initial state r_0 is an antagonist state.
- (2) For each $q \in Q_\psi$ and $V \subseteq \Pi_\psi$ with $q_0^\psi \xrightarrow{V} q \in \delta_\psi$ and each $u \in \tilde{Q}_0^s$ with $\tilde{\pi}(u) = V$, we append the transition $r_0 \rightarrow ((q_0^\psi \xrightarrow{V} q), u)$ to $\bar{\delta}$. Note that the states $((q_0^\psi \xrightarrow{V} q), u)$ are protagonist states.
- (3) For each $((q_1 \xrightarrow{V} q_2), u) \in \delta_\psi \times \tilde{Q}_1$ and transition $\tilde{\tau}(u, \sigma) = u'$ in \tilde{M}_s , we append the transition $((q_1 \xrightarrow{V} q_2), u) \rightarrow ((q_1 \xrightarrow{V} q_2), u')$ to $\bar{\delta}$.
- (4) For each $((q_1 \xrightarrow{V} q_2), u') \in \delta_\psi \times \tilde{Q}_2$ and for each transition $q_2 \xrightarrow{V'} q_3 \in \delta_\psi$ with $V' = \tilde{\pi}(u')$, we append the transition $((q_1 \xrightarrow{V} q_2), u') \rightarrow ((q_2 \xrightarrow{V'} q_3), u')$ to $\bar{\delta}$.
- (5) The set of Büchi states is $\bar{B} = \{((q_1 \xrightarrow{V} q_2), u) \in \delta_\psi \times \tilde{Q}_1 \mid q_2 \in B_\psi\}$.
- (6) The observation of each state is $\bar{\sigma}((q_1 \xrightarrow{V'} q_1), u) = \{u' \in \tilde{Q}_1 \cup \tilde{Q}_2 \mid u' \sim_1 u\}$.

Then clearly the protagonist wins the game $M_{A,\psi}$ if and only if there exists a uniform strategy for the protagonist in \tilde{M}_s such that all paths compatible with that strategy satisfy ψ . The size of $M_{A,\psi}$ is $O(2^{|S|^2 \cdot |Ag| \cdot \log |Act|} \cdot 2^{|\psi|})$.

Finally, we apply results from [25, 51, 53] to transform $M_{A,\psi}$ into a deterministic parity tree automaton as follows:

- (1) Build a two-player parity game with imperfect information \mathcal{G}_{obs} but with an *observable parity* mapping, that is, a parity mapping which is defined on the set of observations (rather than the set of states). The protagonist wins \mathcal{G}_{obs} if and only if it wins $M_{A,\psi}$.
- (2) Build a two-player parity game with perfect information \mathcal{G}_{perf} . Again, the protagonist wins \mathcal{G}_{perf} if and only if she wins \mathcal{G}_{obs} .
- (3) Build a deterministic parity tree automaton whose language is nonempty if and only if the protagonist wins \mathcal{G}_{perf} .

The construction of \mathcal{G}_{obs} can be adapted from [25] (where it is given for parity winning conditions, whereas in our case we only have a Büchi winning condition). First, we consider $M_{A,\psi}$ as a nondeterministic Büchi word automaton N_ψ whose alphabet is the set of pairs consisting of an observation and a label of \mathcal{A}_ψ . We then build a deterministic parity word automaton \mathcal{B}_{det} (using the adaptation of Safra sets from [51]), with the same language as N_ψ . Note that the state space of \mathcal{B}_{det} is in $O(2^{2^{|S|^2 \cdot |Ag| \cdot \log |Act| \cdot 2^{|\psi|}}})$. Also, following [51], the maximal parity in \mathcal{B}_{det} is in $O(2^{|S|^2 \cdot |Ag| \cdot \log |Act| \cdot 2^{|\psi|}})$, hence polynomial in the size of $M_{A,\psi}$.

\mathcal{B}_{det} will then be used to define the acceptance condition of plays in $M_{A,\psi}$ by defining \mathcal{G}_{obs} as the synchronous product with $M_{A,\psi}$, in which the observation mapping \bar{o} is defined as follows: in each pair of states (v, z) with v a state in $M_{A,\psi}$ and z a state in \mathcal{B}_ψ , $\bar{o}(v, z) = z$. Also the parity of each couple (v, z) is defined as the parity of z . This way, the new parity game has an observable parity condition. Then the size of \mathcal{G}_{obs} is in $O(2^{2^{|S|^2 \cdot |Ag| \cdot \log |Act| \cdot 2^{|\psi|}}})$ and the maximal parity in \mathcal{G}_{obs} is in $O(2^{|S|^2 \cdot |Ag| \cdot \log |Act| \cdot 2^{|\psi|}})$.

The game \mathcal{G}_{perf} is then built from \mathcal{G}_{obs} using another adaptation of the information-set construction as in [53]. This construction has two fortunate effects: first, the information-sets are in fact pairs (U, V) with U being a subset of the state-space of $M_{A,\psi}$ and V a state of \mathcal{B}_{det} , due to the fact that this second component is in fact the observation received by the protagonist after each transition. Hence the state space of \mathcal{G}_{obs} is in $O(2^{2^{|S|^2 \cdot |Ag| \cdot \log |Act| \cdot 2^{|\psi|}}})$. Additionally, this does not produce an explosion of the maximal parity of \mathcal{G}_{obs} precisely because the parity is observable (that is, defined on the set of observations), hence still in $O(2^{|S|^2 \cdot |Ag| \cdot \log |Act| \cdot 2^{|\psi|}})$.

The final step is then equivalent with checking nonemptiness of a deterministic parity tree automaton, which is known to require time in $O(m^d)$, where m is the size of the state-space of \mathcal{G}_{perf} and d is the maximal parity therein. Hence we get the double exponential bound in the theorem statement. \square

5.4 Handling nested coalition operators

We may now give the proof of Theorem 8, whose statement we report here.

THEOREM 8. *Given a sequence C of coalitions, the model-checking problem for C -formulas and for iCGS in which each coalition A in C has DS-knowledge is 2 – EXPTIME-complete.*

PROOF. Given an iCGS M and a formula φ , we proceed by adapting the usual iterative process of creating new atoms from subformulas and augmenting the state labelings with the new atoms. W.l.o.g we assume that the leading operator in φ is a coalition operator. At iteration i corresponding with a subformula $\langle\langle A_i \rangle\rangle\psi$ in which ψ does not contain any coalition operator, for each state s we append label $p_{\langle\langle A_i \rangle\rangle\psi}$ to s iff the algorithm in Theorem 14 answers "Yes". We then replace each occurrence of $\langle\langle A_i \rangle\rangle\psi$ with $p_{\langle\langle A_i \rangle\rangle\psi}$ in φ , and iterate with M_{i+1} being the iCGS $M_{\langle\langle A_i \rangle\rangle\psi}$ with the modified labeling. The algorithm stops when treating the whole formula φ itself, in which case we call the algorithm in

Theorem 14 for each subset $E_A(s)$ for some $s \in S_0$. Then our algorithm answers “Yes” iff each of these calls yields a positive answer.

The completeness part follows from the fact that the model-checking problem for ATL^* with perfect information is already $2 - EXPTIME$ -complete. \square

We now describe the text above, i.e., our procedure of checking if a formula ψ holds on a M , in Algorithm 1.

Algorithm 1 Treatment of Formulae with Nested Coalition Operators

Input: formula ψ with nested operators, game structure M

Output: whether formula ψ holds on M

Require: ψ starting with coalition operator

$L^{AP} \leftarrow L_M^{AP}$

$j \leftarrow 0$

$M_j \leftarrow M$

while $\beta \in \text{Subform}(\psi)$ **do**

$X := X_{M_j} \times L^{AP}$

\triangleright pairs of (states in M_j , propositional labels)

$j \leftarrow j + 1$

if $\beta \leftarrow \langle\langle A \rangle\rangle \psi$ **then**

$X := X \cup \{(s, p_\beta) \mid \text{Th}_5(\beta, s) := \text{yes''}\}$

\triangleright Labelling s with new atom p_β

$\psi \leftarrow \text{Transform}(\psi, \beta)$

\triangleright Rewrite ψ by uniformly replacing β with p_β

$L^{AP} \leftarrow L^{AP} \cup \{p_\beta\}$

$M_j \leftarrow \text{Transform}(M_j, X)$

\triangleright Reconstruct a game structure with current X

end if

end while

for $s \in S_{M_j}$ **do**

\triangleright for each state s in the last game structure

if $\text{Th}_5(\psi, E_A(s)) := \text{“No”}$ **then**

\triangleright Call the alg. in Th5 on $E_A(s)$

return “No”

end if

end for

return “Yes”

6 RELEVANCE OF A-CAST ICGS

In this section we illustrate the relevance of the notion of broadcast within a given coalition, or coalition-cast, by showing that it encompasses strictly stronger conditions already considered in the literature on distributed and multi-agent systems. Specifically, we show that coalition-cast iCGSs subsume broadcast systems as defined in [14, 50], as well as systems without information forks [33]. These are all classes of systems that have been proved to allow for a decidable model checking problem under the assumptions of imperfect information and perfect recall.

First, recall from [14] that an iCGS *only has broadcast actions* iff for any $a \in Ag$, states $s_1 \sim_a s_2$ and joint actions $\alpha, \beta \in ACT$, if $\alpha \neq \beta$ then $\tau(s_1, \alpha) \not\sim_a \tau(s_2, \beta)$.

Such a system is *A-cast*: assume that $s_1 \sim_A^D s_2$, $\alpha|_A = \beta|_A$, and $\tau(s_1, \alpha) \sim_A^C \tau(s_2, \beta)$. Then we can prove by induction on the length of the sequence of indistinguishabilities connecting $\tau(s_1, \alpha)$ with $\tau(s_2, \beta)$ that the “only broadcast actions” hypothesis implies $\alpha = \beta$. But then the conclusion follows since $\tau(s_1, \alpha) \sim_A^D \tau(s_2, \alpha)$ by definition.

Recall further from [50] that a *broadcast system* is an iCGS in which

- $Ag = A \cup \{e\}$ with e denoting the “environment” agent.

- $S = \prod_{a \in Ag} S_a$.
- There exists a mapping $O : S_e \longrightarrow Obs$ such that for each $a \in A$ and $\vec{s}_1, \vec{s}_2 \in S$, $\vec{s}_1 \sim_a \vec{s}_2$ iff $\vec{s}_1|_a = \vec{s}_2|_a$ and $O(\vec{s}_1|_e) = O(\vec{s}_2|_e)$.
- For each $\vec{s} \in S$ and $\alpha \in ACT$, $\tau(\vec{s}, \alpha) = \left(\tau_a(\vec{s}|_a, \alpha|_a) \right) \cdot (\tau(\vec{s}|_e, \alpha))$ for some mappings $\tau_a : S_a \times Act_a \longrightarrow S_a$ and $\tau_e : S_e \times ACT \longrightarrow S_e$.

So assume we have $\vec{s}_1 \sim_A^D \vec{s}_2$, which implies that $\vec{s}_1|_a = \vec{s}_2|_a$ for each $a \in A$ and $O(\vec{s}_1|_e) = O(\vec{s}_2|_e)$. Assume further that $\alpha, \beta \in ACT$ with $\alpha|_a = \beta|_a$ for each $a \in A$. It then follows that $\tau_a(\vec{s}_1|_a, \alpha|_a) = \tau_a(\vec{s}_2|_a, \beta|_a)$ for each $a \in A$. If we suppose further that $\tau(\vec{s}_1, \alpha) \sim_b \tau(\vec{s}_2, \beta)$ for some $b \in A$, we then get that $O(\tau(\vec{s}_1, \alpha)|_e) = O(\tau(\vec{s}_2, \beta)|_e)$. As a consequence, $\tau(\vec{s}_1, \alpha) \sim_a \tau(\vec{s}_2, \beta)$ for each $a \in A$. By induction on the length of the sequence of indistinguishabilities connecting $\tau(\vec{s}_1, \alpha)$ and $\tau(\vec{s}_2, \beta)$, we get the A -cast property.

Let us finally comment on the relationship between our notion of RMI without information forks and the notion from [33]. To this end, consider an RMI arena $\mathcal{A} = \langle Ag, Vars, AP, \Phi_I, m_1, \dots, m_{|Ag|} \rangle$, and, for each subset of variables V , define the *agent graph defined by V* as follows: $G_V = (Ag, \xrightarrow{V})$ with $a \xrightarrow{V} b$ iff $V_a \cap Vis_b \cap V \neq \emptyset$. Then the RMI has an *indirect information fork* for agents $a, b, c, d \in Ag$, (in the sense of [33]) if:

- (1) $Ag = A \cup \{e\}$ with e distinct from c and d .
- (2) Φ defines a unique initial state.
- (3) For each set of variables V for which a and b are reachable from e in the graph G_V , with $V \cap Vis_c = \emptyset$, $V \cap Vis_d = \emptyset$, with c and d having no incoming edges in G_V , and with $V_a \cap Vis_c \neq \emptyset$ and $V_b \cap Vis_d \neq \emptyset$, we have that $V_a \cap Vis_c \not\subseteq Vis_d$ and $V_b \cap Vis_d \not\subseteq Vis_c$.

The RMI has a *direct* information fork if condition 3 above applies to $a = b = e$ and some c and d .

We may observe that RMI with visibility control can be seen as a version of RMI in which there exists no agents c and d having a stronger version of direct information forks, but which have multiple initial states, hence the two settings are incompatible. It should be noted that the unique initial state is an essential ingredient in the proof of the decidability from [33], allowing a direct reduction to two-player games with perfect information, and hence avoiding the complexity of solving games with imperfect information and building winning conditions compatible with information sets, as in our approach.

It is not clear whether our technique can be extended to the case of indirect information forks but multiple initial states. We conjecture that this would not be possible, due to results from [50] showing that the realizability problem for specifications in the logic of knowledge and linear time is undecidable in hierarchical systems with two or more agents. The same paper shows that the realizability problem for specifications in which the knowledge operators have only positive occurrences is decidable, which would also suggest that decidability could be obtained for the fragment of ATL^* consisting of positive formulas only, over RMI with indirect information forks and multiple initial states.

7 CONCLUSIONS

In this paper we considered the formalism of vCGS for the explicit representation of private-data sharing in multi-agent systems. On these “MAS with 1-to-1 private-channels”, we introduced coalition-cast where, if agents share private data, then they necessarily share it with all agents in a coalition, and proved verification decidable under the assumptions of perfect recall and imperfect information. We demonstrated how vCGS are amenable to represent collusion by modelling a security-driven example as a vCGS satisfying coalition-cast. Then, we argued that ATL^* is the appropriate language for the security requirement we discussed. In particular, our specification is a first-timer in secure-systems’ verification.

As **future work**, we plan to implement the model-checking algorithm in the MCMAS tool [46]. To date, this tool does not support communication between agents by means of shared variables, the only communication mechanism being synchronisation on joint actions. Therefore, even though our case study can be modelled and verified in MCMAS due to the fact that the model contains a finite number of lasso-type runs, the need to encode shared variables as synchronisation on joint actions would increase even more the complexity of the models, fact which, combined with the necessity to encode agent memory into the model in order to cope with the “memoryless” semantics of ATL, would have a non-negligible impact on the performances of the tool. Therefore, our future research plans imply (1) enriching the syntax of the ISPL to allow for shared variables *and* dynamic visibility of state variables through the addition of *vis* statements, and (2) testing a model for the coalition-cast property, designing and implementing a symbolic version the model-checking algorithm in Section 5.

ACKNOWLEDGMENTS

F. Belardinelli acknowledges the support of the EPSRC Overseas Travel Grant EP/V009214/1 “Strategy Logics for the Verification of Security Protocols”.

REFERENCES

- [1] 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union* L119 (2016), 1–88. <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC>
- [2] 2018. Security of Distance-Bounding: A Survey. *Comput. Surveys* 51, 1–33. <https://doi.org/10.1145/3264628>
- [3] Rajeev Alur and Thomas A. Henzinger. 1999. Reactive Modules. *Formal Methods in System Design* 15, 1 (1999), 7–48. <https://doi.org/10.1023/A:1008739929481>
- [4] R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *J. ACM* 49, 5 (2002), 672–713.
- [5] Francesco Belardinelli, Ioana Boureanu, Catalin Dima, and Vadim Malvone. 2019. Verifying Strategic Abilities in Multi-agent Systems with Private Data-Sharing. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*. 1820–1822. <http://dl.acm.org/citation.cfm?id=3331930>
- [6] F. Belardinelli, R. Condurache, C. Dima, W. Jamroga, and A. V. Jones. 2017. Bisimulations for Verifying Strategic Abilities with an Application to ThreeBallot. In *AAMAS17*. 1286–1295.
- [7] Francesco Belardinelli, Rodica Condurache, Catalin Dima, Wojciech Jamroga, and Michal Knapik. 2021. Bisimulations for verifying strategic abilities with an application to the ThreeBallot voting protocol. *Inf. Comput.* 276 (2021), 104552.
- [8] Francesco Belardinelli, Catalin Dima, Vadim Malvone, and Ferucio Laurentiu Tiplea. 2020. A Hennessy-Milner Theorem for ATL with Imperfect Information. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM, 181–194. <https://doi.org/10.1145/3373718.3394784>
- [9] Francesco Belardinelli, Angelo Ferrando, and Vadim Malvone. 2023. An abstraction-refinement framework for verifying strategic properties in multi-agent systems with imperfect information. *Artif. Intell.* 316 (2023), 103847. <https://doi.org/10.1016/j.artint.2022.103847>
- [10] F. Belardinelli, U. Grandi, A. Herzig, D. Longin, E. Lorini, A. Novaro, and L. Perrussel. 2017. Relaxing Exclusive Control in Boolean Games. In *TARK17*. 43–56.
- [11] Francesco Belardinelli, Alessio Lomuscio, Vadim Malvone, and Emily Yu. 2022. Approximating Perfect Recall when Model Checking Strategic Abilities: Theory and Applications. *J. Artif. Intell. Res.* 73 (2022), 897–932. <https://doi.org/10.1613/jair.1.12539>
- [12] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. 2017. Verification of Broadcasting Multi-Agent Systems against an Epistemic Strategy Logic. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 91–97. <https://doi.org/10.24963/ijcai.2017/14>
- [13] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. 2017. Verification of Multi-agent Systems with Imperfect Information and Public Actions. In *17*. 1268–1276.
- [14] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. 2020. Verification of multi-agent systems with public actions against strategy logic. *Artif. Intell.* 285 (2020), 103302. <https://doi.org/10.1016/j.artint.2020.103302>
- [15] Samy Bengio, Gilles Brassard, Yvo Desmedt, Claude Goutier, and Jean-Jacques Quisquater. 1991. Secure Implementations of Identification Systems. *J. Cryptology* 4, 3 (1991), 175–183. <https://doi.org/10.1007/BF00196726>
- [16] Raphaël Berthon, Bastien Maubert, and Aniello Murano. 2017. Decidability Results for ATL* with Imperfect Information and Perfect Recall. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, Kate Larson,

- Michael Winikoff, Sanmay Das, and Edmund H. Durfee (Eds.). ACM, 1250–1258. <http://dl.acm.org/citation.cfm?id=3091299>
- [17] Raphaël Berthon, Bastien Maubert, Aniello Murano, Sasha Rubin, and Moshe Y. Vardi. 2017. Strategy logic with imperfect information. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 1–12. <https://doi.org/10.1109/LICS.2017.8005136>
 - [18] Dietmar Berwanger and Lukasz Kaiser. 2010. Information Tracking in Games on Graphs. *Journal of Logic, Language and Information* 19, 4 (2010), 395–412. <https://doi.org/10.1007/s10849-009-9115-8>
 - [19] Dietmar Berwanger and Anup Basil Mathew. 2017. Infinite games with finite knowledge gaps. *Inf. Comput.* 254 (2017), 217–237. <https://doi.org/10.1016/j.ic.2016.10.009>
 - [20] B. Blanchet. 2001. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, Nova Scotia, Canada, 82–96.
 - [21] Bruno Blanchet. 2012. Security Protocol Verification: Symbolic and Computational Models. In *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*. 3–29. https://doi.org/10.1007/978-3-642-28641-4_2
 - [22] I. Boureanu and S. Vaudenay. 2015. Challenges in Distance Bounding. *IEEE Security Privacy* 13, 1 (Jan 2015), 41–48. <https://doi.org/10.1109/MSP.2015.2>
 - [23] S. Brands and D. Chaum. 1993. Distance-Bounding Protocols (Extended Abstract). In *EUROCRYPT '93 (Lecture Notes in Computer Science, Vol. 765)*. Springer, 344–359.
 - [24] Xavier Bultel, Sébastien Gambs, David Gérard, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. 2016. A Prover-Anonymous and Terrorist-Fraud Resistant Distance-Bounding Protocol (WiSec '16). Association for Computing Machinery, New York, NY, USA, 121–133. <https://doi.org/10.1145/2939918.2939919>
 - [25] Krishnendu Chatterjee and Laurent Doyen. 2010. The Complexity of Partial-Observation Parity Games. In *Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR) (Lecture Notes in Computer Science, Vol. 6397)*. Springer, 1–14.
 - [26] Alexandre Debant, Stéphanie Delaune, and Cyrille Wiedling. 2018. *Proving physical proximity using symbolic models*. Research Report. Univ Rennes, CNRS, IRISA, France. <https://hal.archives-ouvertes.fr/hal-01708336>
 - [27] Alexandre Debant, Stéphanie Delaune, and Cyrille Wiedling. 2019. Symbolic Analysis of Terrorist Fraud Resistance. In *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part I*. 383–403. https://doi.org/10.1007/978-3-030-29959-0_19
 - [28] C. Dima, C. Enea, and D. P. Guelev. 2010. Model-Checking an Alternating-time Temporal Logic with Knowledge, Imperfect Information, Perfect Recall and Communicating Coalitions. In *Proceedings First Symposium on Games, Automata, Logic, and Formal Verification, Gandalf 2010, Minori (Amalfi Coast), Italy, 17-18th June 2010*. 103–117.
 - [29] C. Dima and F.L. Tiplea. 2011. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. *CoRR* abs/1102.4225 (2011). <http://arxiv.org/abs/1102.4225>
 - [30] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. 1995. *Reasoning about Knowledge*. MIT.
 - [31] Angelo Ferrando and Vadim Malvone. 2022. Towards the Combination of Model Checking and Runtime Verification on Multi-agent Systems. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection - 20th International Conference, PAAMS 2022, L'Aquila, Italy, July 13-15, 2022, Proceedings (Lecture Notes in Computer Science, Vol. 13616)*. Frank Dignum, Philippe Mathieu, Juan Manuel Corchado, and Fernando de la Prieta (Eds.). Springer, 140–152. https://doi.org/10.1007/978-3-031-18192-4_12
 - [32] Angelo Ferrando and Vadim Malvone. 2023. Towards the Verification of Strategic Properties in Multi-Agent Systems with Imperfect Information. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*. Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh (Eds.). ACM, 793–801. <https://doi.org/10.5555/3545946.3598713>
 - [33] Bernd Finkbeiner and Sven Schewe. 2005. Uniform Distributed Synthesis. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*. 321–330. <https://doi.org/10.1109/LICS.2005.53>
 - [34] J. Gerbrandy. 2006. Logics of propositional control. In *AAMAS06*. 193–200. <https://doi.org/10.1145/1160633.1160664>
 - [35] U. Grandi, E. Lorini, A. Novaro, and L. Perrussel. 2017. Strategic Disclosure of Opinions on a Social Network. In *AAMAS17*. 1196–1204.
 - [36] D. Grossi, E. Lorini, and F. Schwarzentruber. 2015. The Ceteris Paribus Structure of Logics of Game Forms. *JAIR* 53 (2015), 91–126. <https://doi.org/10.1613/jair.4666>
 - [37] J. Gutierrez, P. Harrenstein, and M. Wooldridge. 2017. Reasoning about equilibria in game-like concurrent systems. *Ann. Pure Appl. Log.* 168, 2 (2017), 373–403.
 - [38] J. Gutierrez, G. Perelli, and M. Wooldridge. 2016. Imperfect Information in Reactive Modules Games. In *KR'15*. 390–400. <http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12848>
 - [39] Julian Gutierrez, Giuseppe Perelli, and Michael J. Wooldridge. 2018. Imperfect information in Reactive Modules games. *Inf. Comput.* 261 (2018), 650–675. <https://doi.org/10.1016/j.ic.2018.02.023>
 - [40] Gerhard P. Hancke and Markus G. Kuhn. 2005. An RFID Distance Bounding Protocol. In *SecureComm 2005*. 67–73. <https://doi.org/10.1109/SECURECOMM.2005.56>
 - [41] W. Jamroga, M. Knapik, and D. Kurpiewski. 2018. Model Checking the SELENE E-Voting Protocol in Multi-agent Logics. In *Electronic Voting*. Springer International Publishing, 100–116.

- [42] Wojciech Jamroga, Damian Kurpiewski, and Vadim Malvone. 2022. How to measure usable security: Natural strategies in voting protocols. *J. Comput. Secur.* 30, 3 (2022), 381–409. <https://doi.org/10.3233/JCS-210049>
- [43] W. Jamroga and W. van der Hoek. 2004. Agents that Know How to Play. *Fund. Inf.* 62 (2004), 1–35.
- [44] S. C. Kleene. 1952. *Introduction to Metamathematics*. North-Holland.
- [45] O. Kupferman and M. Y. Vardi. 2001. Synthesizing Distributed Systems. In *LICS'01*. IEEE Computer Society, 389–398.
- [46] A. Lomuscio, H. Qu, and F. Raimondi. 2015. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. *Software Tools for Technology Transfer* (2015). <https://doi.org/10.1007/s10009-015-0378-x> <http://dx.doi.org/10.1007/s10009-015-0378-x>.
- [47] F. Maffre. 2016. *Ignorance is bliss : observability-based dynamic epistemic logics and their applications*. Ph.D. Dissertation. Université Paul Sabatier - Toulouse III.
- [48] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. 2018. Distance-Bounding Protocols: Verification without Time and Location. In *S&P 2018*. Springer. <https://doi.org/10.1109/SP.2018.00001>
- [49] S. Meier, B. Schmidt, C. Cremers, and D. Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*. Springer Berlin Heidelberg, Berlin, Heidelberg, 696–701.
- [50] R. van der Meyden and T. Wilke. 2005. Synthesis of Distributed Systems from Knowledge-Based Specifications. In *05*. 562–576.
- [51] Nir Piterman. 2007. From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata. *Log. Methods Comput. Sci.* 3, 3 (2007). [https://doi.org/10.2168/LMCS-3\(3:5\)2007](https://doi.org/10.2168/LMCS-3(3:5)2007)
- [52] A. Pnueli and R. Rosner. 1989. On the Synthesis of a Reactive Module. In *POPL'89*. 179–190.
- [53] Jean-François Raskin, Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. 2007. Algorithms for Omega-Regular Games with Imperfect Information. *Log. Methods Comput. Sci.* 3, 3 (2007).
- [54] R. Chadha, S. Kremer, and A. Scedrov. 2006. Formal Analysis of Multiparty Contract Signing. *J. Autom. Reason.* 36, 1-2 (Jan. 2006), 39–83. <https://doi.org/10.1007/s10817-005-9019-5>
- [55] M. Samaila, M. Neto, D. Fernandes, M. Freire, and P. Inácio. 2017. Security Challenges of the Internet of Things. In *Beyond the Internet of Things: Everything Interconnected*. Springer International Publishing.
- [56] M. Tabatabaei, W. Jamroga, and P. Ryan. 2016. Expressing Receipt-Freeness and Coercion-Resistance in Logics of Strategic Ability: Preliminary Attempt. In *PrAISe@ECAI 2016*. 1:1–1:8.
- [57] W. van der Hoek, A. Lomuscio, and M. Wooldridge. 2006. On the complexity of practical ATL model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS06)*. ACM Press, 201–208.
- [58] W. van der Hoek, D. Walther, and M. Wooldridge. 2010. Reasoning About the Transfer of Control. *Journal of Artificial Intelligence Research (JAIR)* 37 (2010), 437–477. <https://doi.org/10.1613/jair.2901>
- [59] W. van der Hoek and M. Wooldridge. 2005. On the logic of cooperation and propositional control. *Artificial Intelligence* 164, 1-2 (2005), 81–119. <https://doi.org/10.1016/j.artint.2005.01.003>
- [60] H. van Ditmarsch, W. van der Hoek, and B. Kooi. 2007. *Dynamic Epistemic Logic*. Synthese Library, Vol. 337. Springer.

A APPENDIX 1: PROOF OF LEMMA 3

PROOF. Given a uniform strategy σ_A , we define the sequence $(dom^n(\sigma_A))_{n \in \mathbb{N}}$ of sets of histories in \mathbb{M} such that $k \in dom^n(\sigma_A)$ iff k can be reached in at most n steps from some $h \in \chi|_1$ by applying actions compatible with strategy σ_A . Formally, $dom^0(\sigma_A) = C_A(h_0)$ and

$$dom^{n+1}(\sigma_A) = dom^n(\sigma_A) \cup \bigcup_{k \in dom^n(\sigma_A)} \{C_A(l) \mid l \in \text{hext}^2((k, \sigma_A))\}$$

Also, we denote by σ_A^n the partial strategy resulting from restricting σ_A to $dom^n(\sigma_A)$.

We then build $\sigma_{A'}^n$ and a mapping $\theta : out^f(\chi|_2, \sigma_{A'}^n) \rightarrow out^f(\chi|_1, \sigma_A^n)$ such that for all $h \in out_f(\chi|_2, \sigma_{A'}^n)$, $h \Rightarrow \theta(h)$. The construction is done by mutual induction on the length of histories. More formally, we build a sequence of partial strategies $\sigma_{A'}^n : PStr_A(Hist_A^{\leq |h_0|+n}(\mathbb{M})) \rightarrow PStr_{A'}(Hist_{A'}^{\leq |h_0|+n}(\mathbb{M}'))$ for each $n \geq 0$ and a sequence of mappings $\theta^n : out^{n-1}(\chi|_2, \sigma_{A'}^{n-1}) \rightarrow out^{n-1}(\chi|_1, \sigma_A^{n-1})$ as follows. First, for each $h' \in out_{\leq 0}(\chi|_2, \sigma_{A'}^0) = \chi|_2$, we put $\theta^1(h') = rev_{\chi}(h')$. Note that, in this case, we have that, for any $h'_1, h'_2 \in out^0(\chi|_2, \sigma_{A'}^0)$, $\theta^1(h'_1) \sim_A^C \theta^1(h'_2)$ since they both belong to $C_A(h_0)$.

We also define $\sigma_{A'}^0(h') = ST_{C_A(h_0), C_{A'}(h'_0)}(\sigma_A^0)(h')$.

For the induction step, fix a tuple of actions $\alpha'_0 \in Act_{A'}$. Then, for each $n \geq 1$ we assume, by induction, that, whenever $k'_1, k'_2 \in dom(\theta^n)$ with $k'_1 \sim_{A'}^C k'_2$ we also have that $\theta^n(k'_1) \sim_A^C \theta^n(k'_2)$. We call this assumption *compatibility of θ^n with common knowledge for A, A'* . Then, for each $h' \in Hist_A^{\leq |h_0|+n}(\mathbb{M})$, we put $\sigma^n(h') = ST_{C_A(\theta^n(k')), C_{A'}(k')}(\sigma_A)(h')$ if $h' \in C_{A'}(k')$ and $k' \in dom(\theta^n)$, and fix $\sigma^n(h') = \alpha'_0$ otherwise.

Compatibility of θ^n with common knowledge for A, A' implies that, whenever $k'_1 \sim_{A'}^C k'_2$, we have that $ST_{C_A(\theta^n(k'_1)), C_{A'}(k'_1)} = ST_{C_A(\theta^n(k'_2)), C_{A'}(k'_2)}$, which implies that the definition of σ^n above does not depend on the choice of k' . Additionally, by construction, σ^n is uniform on its domain.

We then define $\theta^{n+1} : out^n(\chi|_2, \sigma_{A'}^n) \rightarrow out^n(\chi|_1, \sigma_A^n)$ as follows. Note first that $h' \in out^n(\chi|_2, \sigma_{A'}^n)$ implies that there exists $k' \in out^{n-1}(\chi|_2, \sigma_{A'}^{n-1})$ and $\alpha' \in Act_{A'}$ such that $h' = \text{hext}(k', \sigma_{A'}^{n-1}(k'), \alpha')$. Then we put:

$$\theta^{n+1}(h') = \text{hext}(\theta^n(k'), \sigma_A(\theta^n(k')), \rho_{k, k'}(\alpha'))$$

We need to prove the compatibility of θ^{n+1} with common knowledge for A, A' . To this end, assume some $h'_1, h'_2 \in out^n(\chi|_2, \sigma_{A'}^n)$. Hence for each $i = 1, 2$ there exist $k'_i \in out^{n-1}(\chi|_2, \sigma_{A'}^{n-1})$ and $\alpha'_i \in Act_{A'}$ such that $h'_i = \text{hext}(k'_i, \sigma_{A'}^{n-1}(k'_i), \alpha'_i)$. For simplicity denote $l'_i = \tau'(k'_i, \sigma_{A'}^{n-1}(k'_i))$. By perfect recall, we must also have $k'_1 \sim_{A'}^C k'_2$ and, by the inductive assumption, $\theta^n(k'_1) \sim_A^C \theta^n(k'_2)$. But, since \mathbb{M} is strictly alternating, this implies that $l_1 = \tau(\theta^n(k'_1), \sigma_A(\theta^n(k'_1))) \sim_A^C \tau(\theta^n(k'_2), \sigma_A(\theta^n(k'_2))) = l_2$. Using property (Sim.4) of alternating simulations, we then get that $\tau(l_1, \rho_{l_1, l'_1}(\alpha'_1)) \sim_A^C \tau(l_2, \rho_{l_2, l'_2}(\alpha'_2))$, which is exactly $\theta^{n+1}(h'_1) \sim_A^C \theta^{n+1}(h'_2)$.

Finally, to prove property (*) in the statement of the lemma, consider a path $\lambda' \in out(h', \sigma_{A'}^n)$ and the sequence $(\theta^n(\lambda'_{\leq |h'|+n}))_{n \in \mathbb{N}}$ of histories in \mathbb{M} . By construction, $\theta^{n+1}(\lambda'_{\leq |h'|+n+1}) \in \text{hext}(\theta^n(\lambda'_{\leq |h'|+n}), \sigma_A)$ and $\theta^n(\lambda'_{\leq |h'|+n}) \Rightarrow \lambda'_{\leq |h'|+n}$, which means that this sequence of histories is in fact a path λ in \mathbb{M} which is compatible with σ_A and satisfies $\lambda \Rightarrow \lambda'$, which ends the proof. \square

B APPENDIX 2: PROOF OF THEOREM 13

PROOF. First, note that property (Sim.1) of alternating bisimulations holds by definition. To check property (Sim.2), take a bisimilar pair $[\sigma, h, \beta] \in \hat{Q}_1$ with $[\sigma, h, \beta] \Leftrightarrow_{1,1} (D_{\sigma,h}, \tilde{\sigma}, (h[1], lst(h)), \beta)$. The strategy simulators are defined as follows:

- A. First, for each $\sigma_1 \in PStr_1(C_A([\sigma, h, \beta])) = PStr_A(C_A(h))$, define $\tilde{\sigma}_1 \in PStr_1(C_A(h[1], lst(h)))$ as the following mapping: for each $s_1 s_2 \in C_A(h[1], lst(h))$, $\tilde{\sigma}_1(s_1 s_2) = \sigma(h')$ where h' is some history with $h'[1] = s_1$ and $lst(h') = s_2$. Then

$$ST(\sigma_1) = \tilde{\sigma}_1$$

Remark 4 ensures that, whenever A has DS-knowledge, the definition of $ST(\sigma_1)$ does not depend on the choice of h' .

- B. Dually, for each $\sigma'_1 \in PStr_1(C_A(h[1] \cdot lst(h)))$, define the mapping $\hat{\sigma}'_1 \in PStr_A(C_A(h))$ as follows: for each $h' \in C_A(h)$, $\hat{\sigma}'_1(h') = \sigma'_1(h'[1], lst(h'))$. Then:

$$ST^{-1}(\sigma'_1) = \hat{\sigma}'_1$$

We first list a couple of useful properties that will be used in the sequel:

- (1) For any $h \in Hist(\mathbb{M})$ and $\sigma \in PStr(\mathbb{M})$ for which there exists $\sigma_0 \in Str_A(\mathbb{M})$ such that $\sigma \in PStr_A(\sigma_0, \leq h)$, for each $\beta \in Act_{\bar{A}}$ and each $\sigma_1 \in PStr_A(C_A(\text{hext}(h, \sigma, \beta)))$, we have that

$$\widetilde{\sigma \otimes \sigma_1} = \tilde{\sigma}_1 \quad (4)$$

To see this, take some $ss' \in C_A(h[1] \cdot lst(\text{hext}(h, \sigma, \beta)))$, then for some $h' \in C_A(\text{hext}(h, \sigma, \beta)) \cap out^f(\sigma)$ with $h'[1] = s, lst(h') = s'$, $\widetilde{\sigma \otimes \sigma_1}(ss') = (\sigma \otimes \sigma_1)(h') = \sigma_1(h') = \tilde{\sigma}_1(ss')$, since $h' \in dom(\sigma_1) \setminus dom(\sigma)$.

- (2) Conversely, for any $h \in Hist(\mathbb{M})$ and $\sigma \in PStr(\mathbb{M})$ for which there exists $\sigma_0 \in Str_A(\mathbb{M})$ such that $\sigma \in PStr_A(\sigma_0, \leq h)$, for each $\beta \in Act_{\bar{A}}$ and each $\sigma'_1 \in PStr_1(C_A(h[1] \cdot lst(h)))$, we have that

$$\widetilde{\sigma \otimes \hat{\sigma}'_1} = \sigma'_1 \quad (5)$$

This follows since, take some $h' \in C_A(\text{hext}(h, \sigma, \beta)) \cap out^f(\sigma)$, then $\widetilde{\sigma \otimes \hat{\sigma}'_1}(h') = (\sigma \otimes \hat{\sigma}'_1)(h'[1]lst(h')) = \sigma'_1(h'[1]lst(h')) = \tilde{\sigma}_1(h')$.

Condition (Sim.2) of alternating bisimulations requires proving that, for each $\sigma_1 \in PStr_1(C_A([\sigma, h, \beta])) = PStr_A(C_A(h))$, $\hat{\tau}([\sigma, h, \beta], \sigma_1) \Leftrightarrow_{1,1} \hat{\tau}((D_{\sigma,h}, \tilde{\sigma}, (h[1], lst(h)), \beta), ST(\sigma_1))$. This amounts to proving that:

$$[\sigma \otimes \sigma_1, \text{hext}(h, \sigma, \beta)] \Leftrightarrow_{1,1} (D', ST(\sigma_1), (h[1], s_3))$$

where $s_3 = \tau(lst(h), \tilde{\sigma}(h[1], lst(h)), \beta)$ and $D' = \{(s_4, s_6) \mid \text{for some } (s_4, s_5) \in D_{\sigma,h}, s_4 s_5 s_6 \in \text{hext}^2(s_4 s_5, \tilde{\sigma}) \text{ and } s_6 \sim_C^A s_3\}$. So we need to prove that

- (1) $D_{\sigma \otimes \sigma_1, \text{hext}(h, \sigma, \beta)} = D'$.
- (2) $\widetilde{\sigma \otimes \sigma_1} = ST(\sigma_1)$, which is exactly Identity 4 above.
- (3) $s_3 = lst(\text{hext}(h, \sigma, \beta))$.

The last point follows since $\tilde{\sigma}(h[1], \text{lst}(h)) = \sigma(h)$, and hence $s_3 = \tau(\text{lst}(h), \tilde{\sigma}(h[1], \text{lst}(h)), \beta) = \tau(\text{lst}(h), \sigma(h), \beta) = \text{lst}(\text{hext}(h, \sigma, \beta))$ as per definition of $\text{hext}(h, \sigma, \beta)$ and Identity 2. Finally, the first identity can be proved as follows:

$$\begin{aligned} D_{\sigma \otimes \sigma_1, \text{hext}(h, \sigma, \beta)} &= \{(s_4, s_6) \mid \exists h' \in C_A(\text{hext}(h, \sigma, \beta)) \cap \text{out}^f(\sigma \otimes \sigma_1), h'[1] = s_4, \text{lst}(h') = s_6\} \\ &= \{(s_4, s_6) \mid \exists h'' \in C_A(h) \cap \text{out}^f(\sigma) \text{ with } s_4 = h'[1], \text{ with } s_6 \in \text{lst}(\text{hext}(h'', \sigma)) \\ &\quad \text{and } s_6 \sim_A^C \text{lst}(\text{hext}(h, \sigma, \beta))\} \end{aligned}$$

which is exactly D' if we put $s_5 = \text{lst}(h'')$. Here we have used the fact that $\text{hext}(h'', \sigma) = \text{hext}(h''[1] \cdot \text{lst}(h''), \tilde{\sigma})$.

On the other hand, checking the dual of condition (Sim.2) requires proving that, for each $\sigma'_1 \in PStr_1(C_A(h[1] \text{lst}(h)))$, $\hat{\tau}([\sigma, h, \beta], ST^{-1}(\sigma'_1)) \Leftrightarrow_{1,1} \tilde{\tau}((D_{\sigma, h}, \tilde{\sigma}, (h[1], \text{lst}(h)), \beta), \sigma_1)$. This amounts to proving that:

$$[\sigma \otimes \hat{\sigma}'_1, \text{hext}(h, \sigma, \beta)] \Leftrightarrow_{1,1} (D', \sigma'_1, (h[1], s_3))$$

with the same notations for s_3 and D' as above. So we need to prove that:

- (1) $D_{\sigma \otimes \hat{\sigma}'_1, \text{hext}(h, \sigma, \beta)} = D'$,
- (2) $\sigma \otimes \hat{\sigma}'_1 = \sigma'_1$, which is exactly Identity 5, and
- (3) $s_3 = \text{lst}(\text{hext}(h, \sigma, \beta))$, which was already proved above.

The first identity can be proved almost identically to the above, by simply noting that: $D_{\sigma \otimes \hat{\sigma}'_1, \text{hext}(h, \sigma, \beta)} = D_{\sigma \otimes \sigma_1, \text{hext}(h, \sigma, \beta)}$ for any appropriate one-step strategy $\sigma_1 \in C_A(\text{hext}(h, \sigma, \beta))$, and hence both are equal to D' .

Now let's define the reverse simulation maps:

C. Assume $\hat{h} = [\sigma, h] \Leftrightarrow_{1,1} (D_{\sigma, h}, \tilde{\sigma}(h[1], \text{lst}(h))) = \tilde{h}$, then we set

$$\rho_{\hat{h}, \tilde{h}}(\beta) = \beta \quad \text{and, dually,} \quad \rho_{\tilde{h}, \hat{h}}^{-1}(\beta) = \beta. \quad (6)$$

Property (Sim.3) is immediate, since $\hat{\tau}(\hat{h}, \beta) = [\sigma, h, \beta] \Leftrightarrow_{1,1} (D_{\sigma, h}, (h[1], \text{lst}(h)), \beta) = \tilde{\tau}(D_{\sigma, h}, (h[1], \text{lst}(h)))$.

Finally, for proving condition (Sim.4), assume we are given $\hat{h}_i = [\sigma_i, h_i] \Leftrightarrow_{1,1} (D_{\sigma_i, h_i}, \tilde{\sigma}_i, (h_i[1], \text{lst}(h_i))) = \tilde{h}_i$ (for $i = 1, 2$) with $\hat{h}_1 \sim_1 \hat{h}_2$ and $\tilde{h}_1 \sim_1 \tilde{h}_2$. By construction of \tilde{M} , we must have $D_{\sigma_1, h_1} = D_{\sigma_2, h_2}$ and $\tilde{\sigma}_1 = \tilde{\sigma}_2$. By construction of \hat{M} , we also have $\sigma_1 = \sigma_2$ and $h_1 \sim_A^C h_2$. Assume further some action profiles $\beta_1, \beta_2 \in P(\bar{A}, \text{lst}(h_i))$ such that $\hat{\tau}(\hat{h}_1, \beta_1) \sim_1 \hat{\tau}(\hat{h}_2, \beta_2)$, that is, $[\sigma_1, h_1, \beta_1] = [\sigma_2, h_2, \beta_2]$ (where the indistinguishability is considered in \hat{M}). We must then have $\text{hext}(h_1, \sigma_1, \beta_1) \sim_A^C \text{hext}(h_2, \sigma_2, \beta_2)$, which, in turn, implies that $\text{hext}(h_1[1] \text{lst}(h_1), \sigma_1, \beta_1) \sim_A^C \text{hext}(h_2[1] \text{lst}(h_2), \sigma_2, \beta_2)$. But all these mean that $(D_{\sigma_i, h_i}, \tilde{\sigma}_i, (h_i[1], \text{lst}(h_i)), \beta_1) \sim_1 (D_{\sigma_i, h_i}, \tilde{\sigma}_i, (h_i[1], \text{lst}(h_i)), \beta_2)$, which is the same as the required indistinguishability $\tilde{\tau}(\tilde{h}_1, \beta_1) \sim_1 \tilde{\tau}(\tilde{h}_2, \beta_2)$.

The dual property of (Sim.4) follows similarly: if we assume now that $\tilde{\tau}(\tilde{h}_1, \beta_1) \sim_1 \tilde{\tau}(\tilde{h}_2, \beta_2)$, we must then have $\text{hext}(h_1[1] \text{lst}(h_1), \sigma_1, \beta_1) \sim_A^C \text{hext}(h_2[1] \text{lst}(h_2), \sigma_2, \beta_2)$, which, together with $h_1 \sim_A^C h_2$, implies that $\text{hext}(h_1, \sigma_1, \beta_1) \sim_A^C \text{hext}(h_2, \sigma_2, \beta_2)$, which ensures that $\hat{\tau}(\hat{h}_1, \beta_1) = [\sigma_1, h_1, \beta_1] \sim_1 [\sigma_2, h_2, \beta_2] = \hat{\tau}(\hat{h}_2, \beta_2)$. \square

C APPENDIX 3: LIST OF SYMBOLS

Symbol	Meaning
tt	true
ff	false
Ag	agents

AP	atomic propositions
$Vars$	set of all variables
m_a	module a
V_a	variables controlled by module m_a
Vis_a	variables visible to module m_a
$\mathcal{A} = \langle Ag, Vars, AP, \Phi_I, m_1, \dots, m_{ Ag } \rangle$	RMI arena Φ_I is a formula denoting the initial states
$update_a$	update commands by module m_a
γ	specific update command by module m_a
g	guard g , a Boolean condition over variables
$\gamma := (g \rightsquigarrow v_1 := tv_1, \dots, v_k := tv_k)$	format of update command, where g is a guard
$g(\gamma)$	the guard in a command γ
$asg(\gamma)$	assignment of the guard in γ
φ, ψ	formulae in ATL^*
$\langle\langle A \rangle\rangle$	coalition of a set A of agents
Act_a	the update actions/commands of agent a
S	states in a unwound arena
S_0	initial states in a unwound arena
S_a	local states of agent a
\sim_a	indistinguishability relation of agent a
P	protocol function showing enabled actions at states in an unwound arena
τ	a state-transition function in an unwound arena
$Paths(M)$	the paths of an RMI
$\lambda \in Paths(\dots)$	a generic path λ in an RMI
$M_{\mathcal{A}} = \langle S, S_0, \{Act_a\}_{a \in Ag}, \pi, \{\sim_a\}_{a \in Ag}, P, \tau \rangle$	CGS unwound by RMI arena \mathcal{A}
h	history (a vector/list of states)
f_a , with $f_a : S^+ \rightarrow Act_a$	strategy of agent a
$F_A = \{f_a \mid a \in A\}$	joint strategy for a coalition A of several agents
$out_{sub}(h, F_A)$	outcome of a strategy F_A at history h under subjective interpretation
$out_{obj}(h, F_A)$	outcome of a strategy F_A at history h under objective interpretation
$out(\sigma)$	outcome of a strategy σ , omitting h in $out(h, \sigma)$
\sim_a	knowledge relation for agent a
\sim_A^E	collective knowledge relation for coalition A
\sim_A^C	common knowledge relation for coalition A
\sim_A^D	distributed knowledge relation for coalition A
$\gamma(s)$	state resulting when executing update γ at state s by one agent
$\gamma(s)(v)$	evaluating variable v at the state $\gamma(s)$
$\tau(s, \alpha, \alpha_1)$	means $\tau(s, (\alpha, \alpha_1))$, i.e., actions in α, α_1 are considered together

A -history	history ending in a state with coalition A 's turn
$Hist_A(M)$	set of all A -histories in an iCGS M
$\sigma_a : Hist_A(M) \rightarrow Act_a$	partial (uniform, memoryful) strategy for agent a
$PStr_A(H)$	set of partial strategies whose domain is a set of histories H
$hext(h, \sigma_A)$	the one-step extension of history h with an action provided by strategy σ_A
$out^n(\sigma_A)$	n -step outcomes of a strategy σ_A
$C_A(h)$	neighbourhood with respect to the common knowledge of coalition A over the history h
$ST_{C_A(h), C_{A'}(h')}(\sigma)$ or $ST(\sigma)$	a construction simulating each strategy defined on each history in $C_A(h)$ by a strategy defined on each history in $C_{A'}(h')$
$\Rightarrow_{A,A'}$	a history-based (bi)simulation relations for coalitions for A, A' for turn-based iCGS a la [7]
$PStr_A(\sigma, \leq h)$	partial strategies compatible with some given (full) strategy σ , with domain in the union of all common knowledge neighbourhoods of some prefixes of a history h
$TB(M, A)$	turn-based iCGS based on CGS M and coalition A
\hat{M}	a two-player (infinite state) turn-based game structure in which the protagonist is denoted 1 and the antagonist is 2 (in Section 5.3)

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009