Computationally Feasible Strategies

Catalin Dima
LACL, Université Paris Est Créteil
F-94010 Créteil, France
dima@u-pec.fr

Wojciech Jamroga
Institute of Computer Science, Polish Academy of Sciences
and SnT, University of Luxembourg
wojciech.jamroga@uni.lu

ABSTRACT

Real-life agents seldom have unlimited reasoning power. In this paper, we propose and study a new formal notion of computationally bounded strategic ability in multi-agent systems. The notion characterizes the ability of a set of agents to synthesize an executable strategy in the form of a Turing machine within a given complexity class, that ensures the satisfaction of a temporal objective in a parameterized game arena. We show that the new concept induces a proper hierarchy of strategic abilities – in particular, polynomial-time abilities are strictly weaker than the exponential-time ones. We also propose an "adaptive" variant of computational ability which allows for different strategies for each parameter value, and show that the two notions do not coincide. Finally, we define and study the model-checking problem for computational strategies. We show that the problem is undecidable even for severely restricted inputs, and present our first steps towards decidable fragments.

KEYWORDS

multi-agent systems; strategic ability; bounded rationality; model checking

ACM Reference Format:
Catalin Dima and Wojciech Jamroga. 2023. Computationally Feasible Strategies. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Multi-agent systems (MAS) involve the interaction of multiple autonomous agents, often assumed to exhibit self-interested, goal-directed behavior [47]. Many relevant properties of MAS can be phrased in terms of *strategic abilities* of agents and their groups [2, 11]. In particular, most functionality requirements can be specified as the ability of the authorized agents to achieve their legitimate goals, or to complete their tasks. Moreover, many security properties refer to the inability of the "bad guys" to obtain their goals. For example, a sensible requirement for a coffee vending machine is that a thirsty user can eventually get a cup of the aromatic brew if she follows the right sequence of steps. Similarly, one might want to require that a secure file system disables impersonation, i.e., the intruder cannot log in as another user, no matter how smart strategy he chooses to play.

The "right" semantics of strategic ability in MAS has been a matter of hot debate since early 2000s, cf. e.g. [1, 5, 20, 26, 32, 38, 45]. In particular, it was argued that ability-based requirements should take into account the bounded capabilities of agents [30, 31]. Typically, strategies are defined as functions from sequences of system states (i.e., possible histories of the play) to the agent's choices. The approach is mathematically elegant, and might be appropriate to reason about abilities of agents with unlimited time and reasoning power. However, it makes for a poor model of human behavior, whose strategies should be *easy to obtain*, *easy to remember*, and *efficient to use* [40]; in other words, simple enough to be useful. Similarly, one often wants to look only at threats from *computationally bounded attackers*, for example in cases when unconditional security is unattainable [19, 34].

In this paper, we propose and formalize the concept of *computationally bounded strategic ability*. We draw from two main sources of inspiration. First, it was proposed in [30, 31] to formalize human ability with statements $\langle\langle A\rangle\rangle^{\leq k}\varphi$, saying that agents A have a strategy of *complexity at most* k to achieve goal φ. For instance, one may require that the user has a strategy of complexity at most 10 to get an espresso (think of a sheet of paper with a recipe that uses only 10 symbols or less). Unfortunately, such concrete bounds are usually arbitrary. Why is a recipe of length 10 still good for buying a coffee, but one with 11 symbols is too complex? Why not up to 15 symbols? A possible solution is offered by asymptotic bounds of complexity theory. For a scalable model of the coffee machine, parameterized by the increasing sophistication of the brew it produces, we may demand that the complexity of getting an espresso grows *at most linearly* with the sophistication of the machine.[1] Note that the goal is the same regardless of the sophistication of the machine: getting an espresso.

Secondly, our approach is inspired by cryptographic definitions of security, where the space of potential attack strategies is defined by polynomial-time probabilistic Turing machines [7, 34]. However, with the advances in computing techniques (in particular quantum computers), it is unclear how long polynomial time will remain the boundary separating feasible and unfeasible attacks.

To address this, we propose a general framework that allows us to fix an arbitrary complexity class as the boundary, and ask if there exists a winning Turing-representable strategy within the class, which achieves a fixed objective on all the game structures from an effective class. We consider strategies represented by deterministic Turing machines; the probabilistic case is left for future work. In terms of technical results, we show that different complexity classes induce different strategic abilities, and thus the concept of computationally-bounded strategies gives rise to a new hierarchy of abilities. Moreover, we prove that uniform abilities (i.e., ones where a single general strategy solves all the game instances) do *not* coincide with adaptive abilities (i.e., ones where different strategies can be used for different instances). Finally, we define the model checking problem and show that it is inherently undecidable. We

[1] At most quadratically for Neapolitan users.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

conclude by indicating a couple of decidable cases that can serve as a starting point to obtain more general characterizations in the future.

Related work. Logical formalizations of strategic ability are usually based on alternating-time logic **ATL** [3] or strategy logic **SL** [14, 39]. This includes works discussing different representations and restrictions on strategies, e.g., [20, 27, 32, 38, 45, 51], and establishing the complexity of model checking for various strategy types, cf. [10] for an overview. The models and decidability results for bounded- and finite-memory strategies are of particular relevance [51]. However, none of the frameworks takes into account the complexity of the strategies that the agents need to achieve their objective in the game – with the notable exception of [25] where the complexity of the strategy is considered with respect to the formula size, not the model size, whereas in our case the same objective must be achieved in all models. Complexity bounds have been only used in the natural strategy-based extensions of **ATL** [30, 31] and **SL** [6], and that just in a rudimentary form (concrete rather than asymptotic).

Our research bears a conceptual connection to *bounded rationality* in game theory, where the players have limited computational power when reasoning about their choices [44, 47], e.g., by imposing bounds on their memory [9, 28, 36]. Also, our distinction between uniform and adaptive ability is related to *uniform vs. non-uniform complexity* [12, 23, 46]. Specifically, uniform and non-uniform notions of security were discussed in [8, 17, 24, 35]. Parameterized model checking [4] is also related. The above connections are rather loose; in particular, we define uniformity w.r.t. Turing machines and not Boolean circuits like in the standard approach.

2 PRELIMINARIES

Models of interaction. We consider multi-step games with imperfect information, played by multiple agents over a finite game arena with concurrent synchronous moves [3, 45, 49].

Definition 2.1 (Model). An *imperfect information concurrent game structure (iCGS)*, or simply a *model*, is given by a tuple $M = \langle \mathbb{A}\mathrm{gt}, St, q_0, PV, V, Act, R, t, \{\sim_a\}_{a\in\mathbb{A}\mathrm{gt}}\rangle$ which includes a nonempty finite set of all agents $\mathbb{A}\mathrm{gt} = \{a_1, \dots, a_k\}$, a nonempty finite set of states St, an initial state $q_0 \in St$, a set of atomic propositions PV and their valuation $V : PV \to 2^{St}$, and a nonempty finite set of (atomic) actions Act. The repertoire function $R : \mathbb{A}\mathrm{gt} \times St \to 2^{Act} \setminus \{\emptyset\}$ defines nonempty sets of actions available to agents at each state; we will write $R_a(q)$ instead of $R(a, q)$, and define $R_A(q) = \prod_{a\in A} R_a(q)$ for each $A \subseteq \mathbb{A}\mathrm{gt}, q \in St$. Furthermore, t is a (deterministic) transition function that assigns the outcome state $q' = t(q, \alpha_1, \dots, \alpha_k)$ to each state q and tuple of actions $\langle \alpha_1, \dots, \alpha_k\rangle$ such that $\alpha_i \in R(a_i, q)$ for $i = 1, \dots, k$.

Every $\sim_a \subseteq St \times St$ is an equivalence relation with the intended meaning that, whenever $q \sim_a q'$, the states q and q' are indistinguishable to agent a. The iCGS is assumed to be *uniform*, in the sense that $q \sim_a q'$ implies $R_a(q) = R_a(q')$, i.e., the same choices are available in indistinguishable states. Note that perfect information can be modeled by assuming each $\sim_a$ to be the identity relation.

The *observations* of agent a are defined as $Obs_a = \{[q]_{\sim_a} \mid q \in St\}$, i.e., the equivalence classes of the indistinguishability relation. Assuming an arbitrary ordering of states (e.g., the shortlex ordering, also known as the length-lexicographic ordering), we can uniquely identify each observation $o \in Obs$ with the minimal element of o. Note that the notion of repertoire lifts to observations in a straightforward way.

Strategies and their outcomes. A *strategy* of agent $a \in \mathbb{A}\mathrm{gt}$ is a conditional plan that specifies what a is going to do in every possible situation. The most general variant is given by so called iR *strategies*, i.e., strategies with imperfect information and perfect recall [45]. An iR strategy for a can be formally represented by a function $s_a : {Obs_a}^+ \to Act$ such that $s_a(o_0 \dots o_n) \in R_a(o_n)$.

A collective strategy s_A for coalition $A \subseteq \mathbb{A}\mathrm{gt}$ is simply a tuple of individual iR strategies, one per agent in A.

A *path* $\lambda = q_0 q_1 q_2 \dots$ is an infinite sequence of states such that there is a transition between each q_i, q_{i+1}. We use $\lambda[i]$ to denote the ith position on path λ (starting from $i = 0$) and $\lambda[i, j]$ to denote the part of λ between positions i and j, which includes the case $j = \infty$. Function $out(M, s_A)$ returns the set of all paths that can result from the execution of strategy s_A in model M. For agents outside A, path transitions can involve any actions allowed by their repertoires.

LTL objectives. To specify the "winning condition", i.e. the objective that the coalition wants to achieve, we will use the formulas of *linear time logic* **LTL** [22, 41], built on atomic propositions PV that occur in the models. The syntax of **LTL** is given by the following grammar:

$$\varphi ::= \mathrm{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{X}\,\varphi \mid \varphi\,\mathrm{U}\,\varphi,$$

where $\mathrm{p} \in PV$, X stands for "in the next moment," and U is the temporal operator known as "strong until." The Boolean constants $\bot, \top$ and the other Boolean operators are defined in the standard way. Moreover, the temporal operator "release" can be defined as $\varphi_1\,\mathrm{R}\,\varphi_2 \equiv \neg((\neg\varphi_1)\,\mathrm{U}\,(\neg\varphi_2))$, "sometime in the future" as $\mathrm{F}\,\varphi \equiv \top\,\mathrm{U}\,\varphi$, and "always from now on" as $\mathrm{G}\,\varphi \equiv \bot\,\mathrm{R}\,\varphi$.

The semantics of **LTL** is given by the following clauses:

$\lambda \models_{\mathrm{LTL}} \mathrm{p}$ iff $\lambda[0] \in V(\mathrm{p})$;
$\lambda \models_{\mathrm{LTL}} \neg\varphi$ iff $\lambda \not\models_{\mathrm{LTL}} \varphi$;
$\lambda \models_{\mathrm{LTL}} \varphi_1 \wedge \varphi_2$ iff $\lambda \models_{\mathrm{LTL}} \varphi_1$ and $\lambda \models_{\mathrm{LTL}} \varphi_2$;
$\lambda \models_{\mathrm{LTL}} \mathrm{X}\,\varphi$ iff $\lambda[1, \infty] \models_{\mathrm{LTL}} \varphi$;
$\lambda \models_{\mathrm{LTL}} \varphi_1\,\mathrm{U}\,\varphi_2$ iff $\lambda[i, \infty] \models_{\mathrm{LTL}} \varphi_2$ for some $i \geq 0$ and $\lambda[j, \infty] \models_{\mathrm{LTL}} \varphi_1$ for all $0 \leq j < i$.

We say that strategy s_A enforces objective φ in model M iff $\lambda \models_{\mathrm{LTL}} \varphi$ for every $\lambda \in out(M, s_A)$.

3 COMPUTATIONAL STRATEGIC ABILITY

In this section, we introduce the concept of computational strategies, and show that different complexity bounds produce different views of agents' ability.

3.1 Model Templates

We begin by the introduction of scalable models, in which one can look for a general strategy that wins for all the values of the scalability parameter.

Definition 3.1 (Model template). A *model template* is a countable a family of models $\mathcal{M} = (M_1, M_2, \dots)$. Each model $M \in \mathcal{M}$ is a finite iCGS with the same set of all agents $\mathbb{A}\mathrm{gt}$, actions drawn from

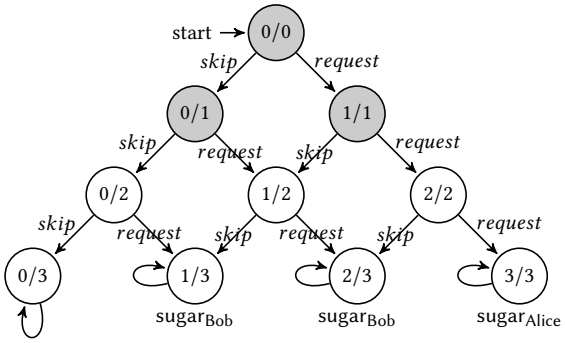

Figure 1: Fibonacci coffee machine M_3^{coffee} for $n = 3$

the same set of actions Act, and atomic propositions drawn from the same set PV.

Typically, $\mathcal{M}$ will be generated by some "security parameter" k with a countable domain of values $\mathcal{K}$. Without loss of generality, we usually assume that $\mathcal{K} = \mathbb{N}$. Operationally, this can be understood as having a procedure that, given a value of k, instantaneously returns the model $M = \mathcal{M}(k)$.

Example 3.2 (Fibonacci espresso machine). Consider a scalable coffee machine that simultaneously provides $n \geq 2$ cups of espresso, numbered $i = 1, 2, \ldots, n$, to n users. The machine has two buttons for controlling the addition of sugar in the espresso cups, with the numbers and the order of the cups that receive sugar being computed as follows: Two designated users (agents), Alice and Bob, need to press one of the two buttons, with Alice doing the first $\lceil \frac{n}{2} \rceil$ choices and Bob the remaining ones. The machine counts the number $m \leq n$ of requests for sugar, then computes the m-th Fibonacci number $F(m)$, then computes its n-bit binary encoding, $F(m) = f_n \ldots f_1 f_0$, and provides the ith user with sugared espresso if and only if $f_{n-i} = 1$. Also Alice gets the last cup of her batch (i.e., the one corresponding to $f_{\lfloor \frac{n}{2} \rfloor}$), and Bob gets the very last one (i.e., for f_0). Atomic propositions $\mathrm{sugar}_{\mathrm{Alice}}$ (resp. $\mathrm{sugar}_{\mathrm{Bob}}$) indicate whether Alice (resp. Bob) get their espresso sweet. Both agents have perfect information about the state of the game.

We can formalize the scenario by a model template $\mathcal{M}_{coffee}$ with $\mathbb{A}\mathrm{gt} = \{Alice, Bob\}$ and $Act = \{request, skip\}$. The states are $St = \{i/j \mid i \leq j, j \leq n\}$, where j indicates the number of decisions already made, and i is the count of requests for sugar. The repertoires are: $R_{Alice}(i/j) = \{request, skip\}$ if $j < \lceil \frac{n}{2} \rceil$ and $R_{Alice}(i/j) = \{skip\}$ otherwise; $R_{Bob}(i/j) = \{request, skip\}$ if $\lceil \frac{n}{2} \rceil \leq j < n$ and else $R_{Bob}(i/j) = \{skip\}$. First, Alice executes a sequence of $\lceil \frac{n}{2} \rceil$ choices, and the state of the model records the count of *request* actions. Then, Bob executes his sequence of $\lfloor \frac{n}{2} \rfloor$ choices, and the counting continues. At the end, coffee is distributed, with the valuation of atomic propositions fixed accordingly. The instance of $\mathcal{M}_{coffee}$ for $n = 3$ is depicted in Figure 1. The states controlled by Alice are set in grey; only the relevant actions are indicated in transition labels.

3.2 Computational Strategies

Strategies are represented by Turing machines that take a history of observations and produce a decision. Strategy templates are Turing machines that take a model and a history, and produce a decision.

Definition 3.3 (Computational strategy). A *computational strategy* s for agent a in model M is an input/output Turing machine with 1 input tape and 1 output tape, that takes as input a sequence of a's observations, and returns as output an action from a's repertoire in the model. We require that the machine is deterministic and total, i.e., terminating on every input.

A *general computational strategy* $\boldsymbol{S}$ for agent a in family $\mathcal{M}$ is an input/output Turing machine with 2 input tapes and 1 output tape, that takes as input the explicit representation of a model and a sequence of a's observations, and returns as output an action from a's repertoire in the model.

If we fix a model $M \in \mathcal{M}$ on the first input tape, the strategy template $\boldsymbol{S}$ gets instantiated to the computational strategy $s = \boldsymbol{S}(M)$.

Example 3.4. The following general strategy of Bob can be easily implemented by a Turing machine of Definition 3.3. Extract the scalability parameter n from the model encoding on the first tape, and take the current state i/j from the history encoding on the second tape. If $j < n$, then write the encoding of *skip* on the output tape. Else, compute $F(i)$ by means of the straightforward recursive algorithm, and write *skip* if the least significant bit of the result is 1, otherwise write *request*.

Remark 1 (Representations of concurrent game structures). *We are going to ask about the existence of winning strategies within a given complexity bound. This might crucially depend on how models and observations are represented as inputs to the Turing machine that implements a strategy. In the rest of this paper, we assume that the elements of* $\mathbb{A}\mathrm{gt}$, Act, *and* St *are ordered in an arbitrary way, and identified by their indices. The indices are encoded in binary, i.e., with appropriate sequences of symbols* $\{0, 1\}$. *Moreover, sets, tuples, and sequences can be represented by enumerating the elements, separated by the special symbol* #, *with the opening and closing brackets encoded as* #00# *and* #01#, *respectively. This way, the representation of an iCGS* M *on the input tape incurs only a logarithmic increase with respect to the "abstract" size of* M, *understood as the total number of states, transitions, and indistinguishability pairs in* M. *Similarly, the representation of a history of observations is at most logarithmically longer than the abstract length of the history.*

Importantly, the Turing machines representing computational strategies use the ternary tape alphabet $\Gamma = \{0, 1, \#\}$.

Definition 3.5 (Collective strategies of coalitions). A computational strategy (resp. general computational strategy) for coalition $A \subseteq \mathbb{A}\mathrm{gt}$ is simply a tuple of computational strategies (resp. general computational strategies), one per agent $a \in A$. We denote the set of computational strategies for A in M by $\mathbf{str}_{M,A}$, and the set of general computational strategies for A in $\mathcal{M}$ by $\mathbf{Str}_{\mathcal{M},A}$.

We use $out(M, s)$ to denote the *outcome set* of strategy s in model M, i.e., the set of infinite paths in M, consistent with s (and analogously for model templates).

Definition 3.6 (Outcome). Given a computational strategy $s \in \mathbf{str}_{M,A}$, we define $out(M, s) = \{\lambda = q_0, q_1, q_2 \ldots \mid$ for each $i = 0, 1, \ldots$ there exists $\vec{\alpha} \in R_{\mathbb{A}\mathrm{gt}}(q_i)$ such that for each $a \in A$, $(\vec{\alpha^i})_a = s_a([q_0]_{\sim_a} \cdot [q_1]_{\sim_a} \cdot \ldots \cdot [q_i]_{\sim_a})$ and $q_{i+1} = t(q_i, \vec{\alpha^i})$.

Moreover, the outcome of a general computational strategy $\boldsymbol{S} \in \mathbf{Str}_{\mathcal{M},A}$ is defined as $out(\mathcal{M}, \boldsymbol{S}) = \bigcup_{M \in \mathcal{M}} out(M, \boldsymbol{S}(M))$.

Example 3.7. The outcome of the strategy $\boldsymbol{S}_{Bob}$, described in Example 3.4, in model M_3^{coffee} of Figure 1 is $out(M_3^{coffee}, \boldsymbol{S}_{Bob}) = \{0/0 \cdot 0/1 \cdot 0/2 \cdot (1/3)^\omega,\ 0/0 \cdot 1/1 \cdot 1/2 \cdot (1/3)^\omega\}$.

The outcome of $\boldsymbol{S}_{Bob}$ in template $\mathcal{M}_{coffee}$ is $out(\mathcal{M}_{coffee}, \boldsymbol{S}_{Bob}) = \{0/0 \dots i/\lceil \frac{n}{2} \rceil \dots i/(n-1) \cdot i/n^\omega \mid n \in \mathbb{N}, 0 \le i \le \lceil \frac{n}{2} \rceil\}$ if $F(i)$ is odd, and $\{0/0 \dots i/\lceil \frac{n}{2} \rceil \dots i/(n-1) \cdot (i+1)/n^\omega \mid n \in \mathbb{N}, 0 \le i \le \lceil \frac{n}{2} \rceil\}$ if $F(i)$ is even.

3.3 Complexity of Models and Strategies

The time complexity of a strategy is defined as the number of steps needed to produce decisions, measured with respect to the size of the input.

Definition 3.8 (Time complexity of general strategies). Consider a model M and a computational strategy s_a of agent a in M. Let $runs(s_a, \alpha)$ denote the set of runs of strategy s_a on the input $enc(\alpha)$, encoding a sequence of observations α. Note that s_a is a Turing machine as per Definition 3.3, and its *runs on* $enc(\alpha)$ should not be confused with its *outcome paths in* M. For a run ρ, we use $|\rho|$ to denote the number of steps on the run. Clearly, $|\rho| \in \mathbb{N}$ if the run is terminating, and $|\rho| = \infty$ otherwise. Similarly, $|enc(\alpha)|$ and $|enc(M)|$ denote the encoding size of the input sequence α and the model M, respectively.

The (pessimistic) *time complexity of general strategy* $\boldsymbol{S}$ is captured by function $time_{\boldsymbol{S}_a} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, defined as follows:

$$time_{\boldsymbol{S}_a}(n,i) = \sup\{|\rho| \mid \rho \in runs(\boldsymbol{S}_a(M), \alpha), M \in \mathcal{M}, |enc(M)| = n, |enc(\alpha)| = i\}.$$

An alternative way to define the complexity of a computational strategy would be to set it against the *abstract* sizes of the model and the history. Note that, under the assumption put forward in Remark 1, such complexity would be at most logarithmically higher than the one defined above.

Definition 3.9 (Complexity of collective general strategies). The *complexity of collective general strategy* $\boldsymbol{S}_A$ is defined as $time_{\boldsymbol{S}_A}(n,i) = \max_{a \in A} time_{\boldsymbol{S}_a}(n,i)$.

Example 3.10. The strategy $\boldsymbol{S}_{Bob}$ of Example 3.4 runs in $O(r^n)$ where r is the *golden ratio*. This is due to the complexity of the straightforward (and naive) computation of Fibonacci numbers [16], used in the strategy.

Note that the definitions in this section can be extended to deterministic *space* complexity in a straightforward way.

3.4 Ability in Computational Strategies

The *uniform* computational ability captures the existence of a computationally feasible strategy template that uniformly wins all the games in $\mathcal{M}$.

Definition 3.11 (Uniform computational ability). Let $\mathcal{M}$ be a model template, φ an **LTL** objective in $\mathcal{M}$, and C a complexity class. Agents $A \subseteq \mathbb{A}\mathrm{gt}$ have *uniform C-ability* in $\mathcal{M}$ for φ (written: $\mathcal{M}, A \models_C \varphi$) if there exists a general strategy $\boldsymbol{S}_A$ for A, such that:

(1) For every path $\lambda \in out(\mathcal{M}, \boldsymbol{S}_A)$, we have that $\lambda \models_{\mathrm{LTL}} \varphi$, and
(2) There exists $f \in C$ such that $\forall n, i\ .\ time_{\boldsymbol{S}_A}(n,i) \le f(n,i)$.

Definition 3.11 can be easily generalized to ω-regular objectives [18, 37].

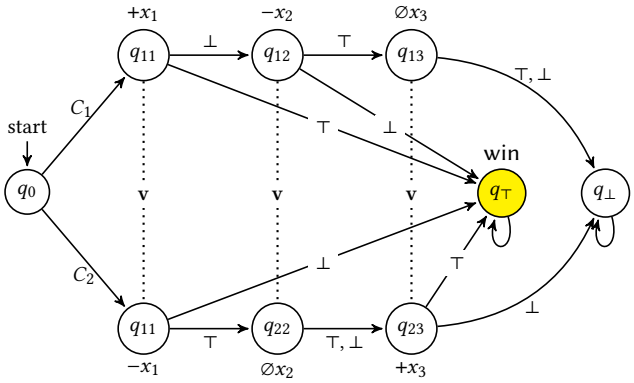

Figure 2: The iCGS M_ϕ for $\phi \equiv (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3)$. The refuter r controls the choice at the initial state q_0, and the verifier v makes all the other choices. We only indicate the action selected by the active player for each transition.

Example 3.12. Going back to our running example, we observe that even the grand coalition cannot uniformly provide both Alice and Bob with sugared espresso, regardless of the complexity bound. This is because Alice's bit of $F(n)$ will always be 0 for large enough values of n. In consequence, $\mathcal{M}_{coffee}, \{Alice, Bob\} \not\models_C \mathrm{F}\,\mathrm{sugar}_{\mathrm{Alice}} \wedge \mathrm{F}\,\mathrm{sugar}_{\mathrm{Bob}}$, for all complexity classes C.

On the other hand, Bob can make sure that he eventually gets his espresso with sugar. For each pair $F(k), F(k+1)$ of subsequent numbers, at least one of them is odd and hence with $f_0 = 1$. Thus, Bob's strategy of Example 3.4 enforces $\mathrm{F}\,\mathrm{sugar}_{\mathrm{Bob}}$. By the observation in Example 3.10, we get that $\mathcal{M}_{coffee}, \{Bob\} \models_{\mathbf{EXPTIME}} \mathrm{F}\,\mathrm{sugar}_{\mathrm{Bob}}$.

Notice further that the strategy can be improved by computing Fibonacci numbers using the recursive algorithm with memoization (i.e., storing the intermediate results), running in time $O(n)$. Thus, we also get $\mathcal{M}_{coffee}, \{Bob\} \models_{\mathbf{P}} \mathrm{F}\,\mathrm{sugar}_{\mathrm{Bob}}$. Finally, if the value of n is encoded explicitly in the representation of M_m^{coffee}, then the strategy can be further optimized to $O(\log n)$ by means of an algorithm based on a matrix representation of the Fibonacci sequence and exponentiation by repeated squaring [16]. In consequence, $\mathcal{M}_{coffee}, \{Bob\} \models_{\mathbf{DLOGTIME}} \mathrm{F}\,\mathrm{sugar}_{\mathrm{Bob}}$.

Observe also that, while Alice cannot make her espresso sweet, she can ensure that she gets it bitter. To this end, it suffices that she always executes *skip*, regardless of what she sees in the input. Thus, $\mathcal{M}_{coffee}, \{Alice\} \models_{O(1)} \mathrm{G}\,\mathrm{sugar}_{\mathrm{Alice}}$.

Remark 2. *In Definition 3.11, we only require the existence of a winning computational strategy for agents A within the given complexity class. We do* not *require that the agent can also computationally* verify *that the strategy is winning and appropriately bounded. This is somewhat analogous to the notions of* objective *vs.* subjective *strategic ability, see [2, Section 11.5.2] for a discussion.*

3.5 Hierarchy of Computational Abilities

We will now show that the concept of ability, proposed in Definition 3.11, is nontrivial. In particular, uniform ability in **P** does not subsume uniform ability in **EXPTIME** (unless **P** =**NP**).

Theorem 3.13. *There exists a model template $\mathcal{M}$, coalition A in $\mathcal{M}$, and* **LTL** *objective φ, such that $\mathcal{M}, A \models_{\mathbf{EXPTIME}} \varphi$ but not $\mathcal{M}, A \models_{\mathbf{P}} \varphi$. This applies even when restricting A to singleton coalitions and φ to reachability objectives.*

Proof. The idea is to take a suitable game encoding of the Boolean satisfiability problem (SAT), and show that (1) the proponent has a general exponential-time strategy to win all the "satisfiable" games, but (2) the existence of a general polynomial-time strategy to do so would imply that SAT can be itself solved in deterministic polynomial time. To this end, we take inspiration from the "Boolean satisfiability games" of [45, Section 3.1] and [29, Section 3.1], which present two different reductions of the SAT problem to model checking of alternating time temporal logic with imperfect information and memoryless strategies. None of the reductions works for strategies with memory, but they can be combined and adapted to our case.

Our encoding of SAT works as follows. Let ϕ be a Boolean formula in Conjunctive Normal Form with n clauses and k Boolean variables $x_1, \ldots, x_k$. We assume w.l.o.g. that the literals in each clause are ordered according to the underlying variable, and literals with the same variable have been reduced. Thus, each clause C_i can be represented as $a_1^i x_1 \wedge \cdots \wedge a_k^i x_k$, with each $a_j^i \in \{+, -, \varnothing\}$ indicating whether C_i includes a positive occurrence of x_j (+), a negated occurrence of x_j (−), or no occurrence of x_j ($\varnothing$).

The corresponding iCGS M_ϕ is constructed as follows. There are two agents, the "verifier" $\mathbf{v}$ and the "refuter" $\mathbf{r}$. First, the refuter chooses a clause in ϕ, without $\mathbf{v}$ knowing which one has been chosen. Then, the verifier goes through the literals in the clause, one by one, declaring the value of the underlying proposition to be either $\top$ or $\bot$. If it makes the literal true, the game proceeds to the winning state $q_\top$, otherwise the system moves to the next literal in the clause (literals with $\varnothing$ are unsatisfiable by definition). If this has been the last literal in the clause, then the game ends in the losing state $q_\bot$. At each stage, the verifier only knows which literal in the clause (equivalently, which Boolean variable) is currently handled. As an example, the satisfiability game for $\phi \equiv (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3)$ is presented in Figure 2. Similarly to [29, 45], we have that

ϕ is satisfiable iff there exists an iR strategy for $\mathbf{v}$ in M_ϕ that surely obtains F win. (*)

Clearly, the winning iR strategy can be always found by a brute-force search checking all iR strategies in M_ϕ in exponential time w.r.t. the size of M_ϕ. (**)

Take $\mathcal{M}_{Sat}$ to be the set of satisfiability games M_ϕ for all the satisfiable formulas ϕ, let $\mathcal{M}_{Unsat}$ be the set of satisfiability games for the unsatisfiable formulas, and $\mathcal{M}_{Bool} = \mathcal{M}_{Sat} \cup \mathcal{M}_{Unsat}$ the set of satisfiability games for all the Boolean formulas. Clearly, each of those is a countable set of finite iCGS's; we assume the shortlex ordering of their elements when needed. By (*) and (**), we get $\mathcal{M}_{Sat}, \{\mathbf{v}\} \models_{\mathbf{EXPTIME}} \mathrm{F}\,\mathsf{win}$.

Suppose that $\mathcal{M}_{Sat}, \{\mathbf{v}\} \models_{\mathbf{P}} \mathrm{F}\,\mathsf{win}$ is also the case, i.e., the verifier has a polynomial-time general strategy $\boldsymbol{S}_\mathbf{v}$ that obtains F win in $\mathcal{M}_{Sat}$. We will show that $\boldsymbol{S}_\mathbf{v}$ can be used to construct a deterministic polynomial-time algorithm to solve Boolean satisfiability, thus contradicting **NP**-completeness of the latter problem (unless **P** = **NP**). The algorithm proceeds as follows:

(1) Given a Boolean formula ϕ, construct its sat. game M_ϕ;
(2) Generate the prefixes of length $k + 2$ for all the runs of $\boldsymbol{S}_\mathbf{v}$ in M_ϕ. If all end up in $q_\top$, return *true*; otherwise, return *false*.

Note that:

- By assumption, $\boldsymbol{S}_\mathbf{v}$ runs in polynomial time for any input over alphabet $\{0, 1, \#\}$, in particular for the representation of any $M_\phi \in \mathcal{M}_{Bool}$ stored on the first input tape. However, it is only guaranteed to implement a *winning* strategy for inputs representing iCGS's in $\mathcal{M}_{Sat}$. In fact, we know that such strategies for $\mathcal{M}_{Unsat}$ cannot exist (and thus cannot be represented).
- The outcome of any $\mathbf{v}$'s strategy in M_ϕ contains exactly n paths, one for each possible choice of the refuter in q_0.
- Each path in M_ϕ reaches either $q_\top$ or $q_\bot$ in exactly $k + 2$ steps, and stays there forever from then on.

We claim that the procedure runs in polynomial time, and returns *true* iff ϕ is satisfiable, which obtains the contradiction and completes the proof. □

4 ADAPTIVE COMPUTATIONAL STRATEGIES

Uniform security notions are rather weak in the sense that the intruder must have a *single* attack strategy that works for all values of the security parameter (e.g., all the possible key lengths). If such a powerful attack strategy does not exist, the system is deemed secure. In real life, we should be equally worried by the possibility that the intruder always has an efficient attack strategy against the *current* value of the parameter. When we upgrade the security parameter to disable the attack, he takes some time to prepare a new attack strategy that efficiently works against the new security parameter, and so on. We formalize this kind of strategic abilities here. We note in passing the relation to non-uniform complexity theory [12]. However, unlike standard approaches that use Boolean circuits and measure complexity in terms of the circuit size or depth [12, 23, 46], we build directly on the time complexity of Turing machines.

4.1 Non-Uniform Strategies and Abilities

Complexity bounds are by definition asymptotic. How can we define asymptotic bounds when considering a different strategy for each single model? The solution is to look at *collections* of such strategies, one per model. In line with that, we generalize Definitions 3.8 and 3.9 to any mapping from a family of models to strategies in those models.

Definition 4.1 (Strategy mappings). $ST_A : \mathcal{M} \to \mathbf{str}_{\mathcal{M},A}$ is a *strategy mapping for A* if $ST_A(M) \in \mathbf{str}_{M,A}$ for every model $M \in \mathcal{M}$. Intuitively ST_A represents a family of computational strategies, each of them fitted for a different game that the agents in A might come to play. The outcome of a strategy mapping is defined as $out(\mathcal{M}, ST) = \bigcup_{M \in \mathcal{M}} out(M, ST(M))$.

Definition 4.2 (Complexity of strategy mappings). The time complexity of strategy mapping ST_A is defined as

$$\begin{aligned} time_{ST_a}(n, i) &= \sup\{|\rho| \mid \rho \in runs(ST_a(M), \alpha), \\ &\qquad M \in \mathcal{M}, |M| = n, |\alpha| = i\} \\ time_{ST_A}(n, i) &= \max_{a \in A} time_{ST_a}(n, i); \end{aligned}$$

where $|M|$ is the abstract size of model M, measured by the total number of states, transitions, and indistinguishability links in M.

Definition 4.3 (Adaptive computational ability). Let $\mathcal{M}$ be a family of models, φ an **LTL** objective in $\mathcal{M}$, and C a complexity class. Agents $A \subseteq \mathbb{A}\text{gt}$ have *adaptive C-ability* in $\mathcal{M}$ for φ (written: $\mathcal{M}, A \models_C \varphi$) if there exists a strategy mapping ST_A for A in $\mathcal{M}$, such that:

(1) For every path $\lambda \in out(\mathcal{M}, ST_A)$, we have that $\lambda \models_{\text{LTL}} \varphi$, and
(2) There exists $f \in C$ such that $\forall n, i \,.\, time_{ST_A}(n, i) \leq f(n, i)$.

4.2 Uniform vs. Adaptive Ability

In this section we prove the main technical result of this paper, stating that uniform and adaptive abilities do not coincide. In fact, uniform ability is strictly stronger than adaptive ability.

Proposition 4.4. *Uniform ability always implies adaptive ability. Formally, for every model template $\mathcal{M}$, coalition A in $\mathcal{M}$, complexity class C, and* **LTL** *objective φ, if $\mathcal{M}, A \models_C \varphi$ then $\mathcal{M}, A \models_C \varphi$.*

Proof. Straightforward. □

Theorem 4.5. *Adaptive ability does not universally imply uniform ability. Formally, there exists a model template $\mathcal{M}$, coalition A in $\mathcal{M}$, complexity class C, and* **LTL** *objective φ, such that $\mathcal{M}, A \models_C \varphi$ but not $\mathcal{M}, A \models_C \varphi$. This applies even when restricting A to singleton coalitions and φ to reachability objectives, and fixing $C = \mathbf{P}$.*

Proof. We take the game encoding $\mathcal{M}_{Sat}$ of the SAT problem from the proof of Theorem 3.13. It was already shown there that $\mathcal{M}_{Sat}, \{\mathbf{v}\} \not\models_{\mathbf{P}} \mathsf{F}\,\mathsf{win}$. It remains to prove that $\mathbf{v}$ has a collection of polynomial-time strategies to win each game in $\mathcal{M}_{Sat}$. Recall that

$$\phi \text{ is satisfiable} \quad \text{iff} \quad \text{there exists an iR strategy for } \mathbf{v} \text{ in } M_\phi \text{ that surely obtains } \mathsf{F}\,\mathsf{win}. \ (*)$$

Moreover, each iR strategy of $\mathbf{v}$ can be represented explicitly by a mapping from the number of observations in the history (which is bounded by $k + 2$) to actions in $\{\bot, \top\}$, thus also by a deterministic Turing machine with $k + 2$ states, counting the number of observations on the input tape, and writing the appropriate decision as output. Clearly, the machine runs in time polynomial w.r.t. k. (**) By (*) and (**), we have that $\mathcal{M}_{Sat}, \{\mathbf{v}\} \models_{\mathbf{P}} \mathsf{F}\,\mathsf{win}$, i.e., the verifier has adaptive polynomial-time ability for $\mathsf{F}\,\mathsf{win}$ in $\mathcal{M}_{Sat}$. □

The proof of Theorem 4.5 relies on the assumption that $\mathbf{P} \neq \mathbf{NP}$ and on the requirement that computational strategies must terminate *on any input* (and not only the inputs that encode the game models in $\mathcal{M}$). We conjecture that the former assumption is essential, but the latter is not. Finding an alternative proof that does not employ the strong termination requirement is a matter for future work.

5 MODEL CHECKING FOR COMPUTATIONAL STRATEGIES

In this section, we formally define the problem of model checking for computationally bounded strategies, and study its decidability. According to Rice's theorem, any non-trivial semantic property of Turing machines[2] is undecidable [43]. Since we restrict computation strategies to *terminating* Turing machines, our framework is not directly applicable for Rice's theorem. Still, one should expect mostly bad news here.

5.1 Problem Definition

First, we formally define what it means to verify the computational ability of a coalition in a given family of game models. As before, we assume a finite nonempty set of all agents $\mathbb{A}\text{gt}$, a nonempty set of actions Act, and a countable set of atomic propositions PV.

Definition 5.1 (Model checking). Model checking of uniform computational abilities is defined as the following decision problem.
Input:

- A specification of a countable family of iCGS's $\mathcal{M} = (M_1, M_2, \dots)$, given by a terminating Turing machine *gen* with 1 input tape and 1 output tape; given $k \in \mathbb{N}$ as input, it generates a representation of the iCGS M_k as the output.
- An **LTL** formula φ, possibly using the encodings of atomic propositions in $PV_{\mathcal{M}}$, defined by *gen*;
- A coalition $A \subseteq \mathbb{A}\text{gt}$;
- A complexity class C.

Output: *true* if $\mathcal{M}, A \models_C \varphi$, otherwise *false*.

Model checking of adaptive computational abilities can be defined analogously, with the output being *true* if $\mathcal{M}, A \models_C \varphi$, and *false* otherwise.

5.2 Bad News: Undecidability

The problem is undecidable even in the following special case.

Theorem 5.2. *Model checking for uniform computational abilities is undecidable even for singleton coalitions with safety objectives and any complexity constraint from $O(n)$ upwards.*

Proof. We prove the undecidability by a reduction of the nonhalting problem for deterministic Turing machines with initially empty input tape. Given a representation of such a Turing machine TM, we construct the model template $\mathcal{M}_{TM}$ as the set of all the finite prefixes of the sole run of TM, with $\mathbb{A}\text{gt} = a$, all the transitions labelled by the sole action label *idle*, and the accepting configurations of TM labelled by the sole atomic proposition accepting. The template is Turing-representable in a straightforward way. Note that there is only 1 strategy available in $\mathcal{M}_{TM}$, namely the automatic strategy that returns *idle* regardless of the input. Moreover, the automatic strategy has complexity $O(1)$.

Then, TM does not halt iff the automatic strategy enforces $\mathsf{G}\,\neg$accepting iff $\mathcal{M}_{TM}, \{a\} \models_{O(1)} \mathsf{G}\,\neg$accepting, which completes the reduction. □

5.3 Further Bad News

The undecidability result in Section 5.2 relied on the fact that $\mathcal{M}$ was a countable family of models. Here, we show that the problem can be undecidable even for a single game. Note that the uniform and adaptive abilities coincide when $\mathcal{M}$ is a singleton, hence the proof below covers both variants of model checking.

[2] A property is semantic if it depends only on the function computed by the machine. It is nontrivial if there are both machines that satisfy it, and ones that do not.

Theorem 5.3. *Model checking for computational abilities is undecidable even for coalitions of size 2 with safety objectives and polynomial-time complexity constraints over model templates that consist of a single game.*

Proof. We recall here the construction from [21]. Namely, given a deterministic TM M which starts with a blank tape, we build a 3-agent iCGS whose set of atomic propositions is a singleton $\{ok\}$ such that $\langle\langle\{1,2\}\rangle\rangle\mathbb{G}\, ok$ iff M halts. We then show that the winning strategy for the coalition can be implemented by a polytime TM.

The informal description of the iCGS is as follows. Initially, agent 3 (the adversary) chooses a tape cell c_k. Then, agents 1 and 2 must simulate the evolution of the tape cell c_k on the unique run ρ_M of M. By the evolution of a tape cell up to some time instant, we mean the sequence consisting of the alphabet symbol that the respective cell contains, plus an information bit about whether the R/W head points to cell c_k or not. The 3rd agent may also choose the "frontier" between the two cells c_k and c_{k+1} in order to test that the two agents correctly simulate the exact moments when the R/W head moves from c_k to c_{k+1} or the other way round – in the sense to be explained below. In order to cope with a fixed number of choices, agent 3's choices are staggered among several steps, as in [21].

The evolution of tape cell c_k in ρ_M up to time instant n is encoded by a sequence of $2n+4$ iCGS states which may be of two types: the tape symbol a where a is the contents of c_k at some $i \leq n$ configuration $\rho_M[i]$ and the R/W head points to another cell, or the pair (a, q) where a is the contents of c_k and the R/W head points to c_k at $\rho_M[i]$. Simulating the $(n+1)$-th TM transition (that is, $\rho_M[n+1]$) on such a sequence of states of length $2n+4$ (call it a history of length $2n+4$) increases the length of the sequence by 2 new states, as explained in the following:

In a history of length $2n+3$ and ending in a, both agents should play *idle* during the following 2 rounds if c_k is not modified by the n-th transition in ρ_M and the R/W head is not moving to c_k. On the contrary, when in the n-th transition of ρ_M, the R/W head "**moves right** and reaches c_k" by applying some transition (q, b, r, b', R), agent 2 must play action (q, r, R) while agent 1 must play *idle*. The state (a, r) is appended to the history (which has now length $2n+4$). Dually, when the R/W head "**moves left** and reaches c_k" by applying some transition (q, b, r, b', L), agent 1 plays (q, r, L) while agent 2 plays *idle*, state (a, r) is appended to the history. Subsequently, if the $(n+1)$-th transition in ρ_M is (r, a, s, c, L), agent 2 should choose action (r, s, L), while agent 1 should play *idle*. The target state of this joint action is c, to account for the fact that c_k holds a c. A dual situation occurs when the $(n+1)$-th transition in ρ_M is (r, a, s, c, R), agent 1 should choose action (r, s, R), while agent 2 should play *idle*, and the target state is c.

When the 3rd agent challenges the two agents with correctly simulating the evolution of the "frontier" between c_k and c_{k+1}, the evolution of this "frontier" in ρ_M up to time instant n is simulated by another type history of length $2n+4$, composed by either a special state tr (transitory) or some triples (q, r, L) or (q, r, R) with q, r being states of the iCGS. Their meaning is as follows:

When the history ends in tr and the frontier is not crossed during the n-transition $\rho_M[n]$, the two agents should both play *idle* during the following 2 rounds and two states tr are then appended to the history at positions $2n+5$ and $2n+6$. When the frontier is crossed "**from right to left**", due to the application of some transition (q, a, r, b, L), agent 2 should play (q, r, L), while agent 1 should play *idle*, and a new state (q, r, L) is appended to the history. Subsequently, agent 1 should immediately issue the action (q, r, L) while agent 2 plays *idle* and another tr state is appended to the history. The dual situation of crossing the frontier "**from left to right**", due to the application of some transition (q, a, r, b, R), is simulated by agent 1 playing (q, r, R), while agent 2 should play *idle*, and the state (q, r, R) is appended in position $2n+5$. Subsequently, in (q, r, R), agent 2 plays (q, r, R) while agent 1 plays *idle* and state tr is appended in position $2n+6$.

Indistinguishability is used to verify that the evolution of two adjacent tape cells and their frontier is correctly simulated by the two protagonists. Hence, agent 1 does not distinguish a history which corresponds with the evolution of tape cell c_k upto some configuration n in the TM run from a history of the same length which corresponds with the evolution of the frontier between c_k and c_{k+1} upto configuration n. Similarly, agent 2 does not distinguish a history which corresponds with the evolution of tape cell c_{k+1} upto configuration n from a history of the same length which corresponds with the evolution of the frontier between c_k and c_{k+1} upto configuration n. Therefore, when agent 1 plays action (q, r, L) on a history corresponding with the evolution of tape cell c_k upto instant n, she must play the same action on the history corresponding with the evolution of the frontier between cells c_{k-1} and c_k.

To enforce the fact that a "good" joint strategy must correspond with a nonblocking run, all transitions with combinations of actions not listed above lead to an error state labeled $\neg ok$. An example is a joint transition with label $((q, r, L), idle)$ from a state labeled (a, s) with $s \neq q$ – the "good" joint transition should be $((s, r, L), idle)$, and only if (s, a, r, b, L) is a legal TM transition. As a consequence, when the TM M halts, say, with the R/W head pointing c_k which bears symbol a and the current state being s, all joint transitions from (a, s) in the iCGS lead to the error state. So, in order to avoid $\neg ok$ states, the protagonists must ensure that they correctly simulate the good transitions on each "cell" or "frontier" history, and each such history can be unboundedly extended, that is, the TM M has a nonblocking run.

We now show that, when the TM M never halts, the unique joint strategy σ described above can be implemented by a polytime TM T_σ. Let us note first that the decisions that each agent must make are based on the decomposition of the history in two parts:

(1) A part of length $O(k)$ which identifies the index of the simulated tape cell c_k, or the frontier between c_k and and c_{k+1}.
(2) A part of length $O(n)$ which identifies the length of ρ_M at the end of which the two agents must decide how the tape cell c_k or the frontier between c_k and c_{k+1}.

Then T_σ must simulate, on an internal tape, the first n transitions in the unique run ρ_M, then read the contents of the k-th tape symbol plus the information whether the R/W head of M crosses one of the frontiers of c_k, and produce the appropriate decision for each agent. During this simulation, only the first n tape symbols need to be simulated, since the R/W head cannot visit more than n cells during the first n steps in ρ_n. Each simulation of one transition of M requires time polynomial in n, plus the generation of the n blank tape cells of the initial configuration, requiring again time

polynomial in n, plus collecting the information for the k-th tape cell and/or its neighbor, requiring again time polynomial in n. □

5.4 In Quest for Decidable Fragments

Theorems 5.2 and 5.3 show that the problem is inherently undecidable. That was hardly unexpected. In fact, we are aware of no nontrivial decidability results for problems that answer if a particular subclass of instances of a more general decision problem can be solved in a given complexity class. Even deciding whether an arbitrary subset of Boolean satisfiability (SAT) can be tackled in deterministic polynomial time has been elusive so far [48].

In this subsection, we list a couple of decidable cases for model checking of computational ability. All of them follow from "complexity saturation" properties, i.e., if a winning strategy exists at all, it must be computable within the given complexity bound. In this sense, the results are not extremely exciting. We hope, however, that the insights can serve as a starting point to obtain more interesting characterizations in the future.

Theorem 5.4. *Model checking for uniform computational abilities in case of singleton coalitions $A = \{a\}$, singleton families of games $\mathcal{M} = \{M_l\}$, and complexity constraints from $O(n)$ up is decidable.*

Proof. We transform the iCGS M_l into a two-player game with perfect information and a parity condition in which the states are labeled with observations of the agent a, such that a memoryless strategy for agent a exists in M_l if and only if the protagonist wins the two-player game. For that, we first build a deterministic parity automaton $\mathcal{A}$ which encodes the LTL formula, then compute the synchronous composition of M_l and $\mathcal{A}$ to obtain a two-player game with imperfect information and parity winning condition. Then we utilize results from [13, 42] to transform this game into a parity game with perfect information with the desired property and which the protagonist wins if and only if she wins with a memoryless strategy, that is, a strategy in which the choice of an action depends only on the last state of the history.

Then the Turing machine for the general strategy S_a for a would have, as its set of states, the states of the two-player game, and, for each sequence of observations, would output the action prescribed by the winning strategy in the two-player game at the unique state of that game occurring at the end of the sequence of observations. Note that this Turing machine works in time linear in the length of the sequence of observations. Note also that the size of the two-player game is at most quadruply-exponential in the size of the original game structure and the LTL formula, but, since the size of the input model is treated as constant, this has no impact on the complexity of the winning strategy. □

For the next theorem we adapt the notion of multi-energy games from [15, 33, 50] in order to extract an interesting subproblem of model-checking. An *iCGS with n counters* is an iCGS M in which all the components of M are the same as in Definition 2.1, but transitions are tuples of the form $(q, \overline{\alpha}, \gamma, v, r)$ where:

- $\overline{\alpha} \in Act^{\mathbb{A}\mathrm{gt}}$, γ is a *positive* boolean combination of formulas of the type $c_i > 0$, where c_i is one of a fixed set of n counter variables.
- v is a set of operations of the type $(++c_i)$ or $(- -c_i)$.

The "semantics" of the n-counter iCGS is an infinite iCGS whose states – called "configurations" – consist of a state and a tuple of n counter values. A transition labeled by $\overline{\alpha}$ exists between two configurations $K_1 = (q_1, c_1, \ldots, c_n)$ and $K_2 = (q_2, c'_1, \ldots c'_n)$ if there is a transition $(q_1, \overline{\alpha}, \gamma, v, q_2)$ such that $(c_1, \ldots, c_n) \models \gamma$ and $(c'_1, \ldots, c'_n)$ is obtained from $(c_1, \ldots, c_n)$ by applying operations in v.

Given two integers N_0 and k and a multi-energy model M, the *(N_0, k)-bounded model* $M_{N_0,k}$ is obtained by bounding the possible values of $c_1, \ldots, c_n$ by $N_0 + k$ in all configurations. Additionally, an attempt to increment the value of some counter to more than $N_0 + k$ results in no change of the counter value.

The class of *energy-bounded families of iCGS* generated by a multi-energy model M and an integer N_0 is the class $\mathcal{M} = (M_k)_{k \in \mathbb{N}}$ where $M_k = M_{N_0,k}$ is defined as above.

Theorem 5.5. *Model checking computational abilities in energy-bounded families of iCGS and singleton coalitions is decidable.*

Proof. Note that, to check the existence of a strategy for agent a which enforces an LTL formula for all M_k, it suffices to check if there is a strategy that enforces the formula in the model M_0, since each sequence of transitions which is feasible for N_0 is also feasible for a larger upper bound. Furthermore, checking the existence of a strategy for M_0 requires building the finite-state two-player game arena from M_0, and then utilizing again the classical results on constructing parity automata from LTL formulas and [13, 42] for making parities visible and determinizing the resulting automaton. As a result, we obtain a two-player finite-state parity game which can be won by the protagonist if and only if in the original iCGS M_0, player a would have a strategy to enforce the given LTL formula. Again, this would require a memoryless strategy which provides a general strategy implementable by a Turing machine working in linear time, as in the proof of Theorem 5.4. □

6 CONCLUSIONS

We present the concept of computationally-bounded strategic ability, which defines the agents'power in terms of their ability to synthesize a winning algorithm working within a given complexity bound, which computes their choices in a parameterized game arena. Our notion is inspired by cryptographic definitions of security, research on human-friendly strategies, and parameterized model-checking. We show that the hierarchy of computational abilities does not collapse, since polynomially-bounded strategies are strictly weaker than exponentially-bounded ones. Moreover, we show that the uniform and non-uniform variants of computational ability do not coincide. We also define a class of model checking problems which can be thought of as a computationally parameterized variant of strategy synthesis for multi-agent systems with imperfect information. We show that the problem is undecidable even for very restricted classes of inputs, and provide some simple cases in which the problem becomes decidable.

For future work, we plan to investigate in closer detail the connection with cryptographic definitions of security, and to identify more relevant decidable classes of the model-checking problem, in which the hierarchy of computational strategic abilities does not collapse above linear time. The complexity of the model-checking problem for the decidable cases is another interesting path of future research.

ACKNOWLEDGMENTS

The work has been supported by NCBR Poland and FNR Luxembourg under the PolLux/FNR-CORE projects STV (POLLUX-VII/1/2019 & C18/IS/12685695/IS/STV/Ryan) and SpaceVote (POLLUX-XI/14/SpaceVote/2023).

REFERENCES

[1] T. Ågotnes. 2004. A note on Syntactic Characterization of Incomplete Information in ATEL. In *Procedings of Workshop on Knowledge and Games*. 34–42.
[2] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge. 2015. Knowledge and Ability. In *Handbook of Epistemic Logic*, H.P. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B.P. Kooi (Eds.). College Publications, 543–589.
[3] R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *J. ACM* 49 (2002), 672–713. https://doi.org/10.1145/585265.585270
[4] Benjamin Aminof, Aniello Murano, Sasha Rubin, and Florian Zuleger. 2016. Automatic Verification of Multi-Agent Systems in Parameterised Grid-Environments. In *Proceedings of AAMAS*. ACM, 1190–1199.
[5] B. Aminof, A. Murano, S. Rubin, and F. Zuleger. 2016. Prompt Alternating-Time Epistemic Logics. In *Proceedings of KR*. 258–267.
[6] Francesco Belardinelli, Wojtek Jamroga, Vadim Malvone, Munyque Mittelmann, Aniello Murano, and Laurent Perrussel. 2022. Reasoning about Human-Friendly Strategies in Repeated Keyword Auctions. In *Proceedings of AAMAS*. IFAAMAS, 62–71. https://doi.org/10.5555/3535850.3535859
[7] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. 1998. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology - Proceedings of CRYPTO*. 26–45. https://doi.org/10.1007/BFb0055718
[8] Daniel J. Bernstein and Tanja Lange. 2013. Non-uniform Cracks in the Concrete: The Power of Free Precomputation. In *Proceedings of ASIACRYPT, Part II (Lecture Notes in Computer Science, Vol. 8270)*. Springer, 321–340. https://doi.org/10.1007/978-3-642-42045-0_17
[9] V. Bhaskar, G.J. Mailath, and S. Morris. 2012. A foundation for Markov equilibria in sequential games with finite social memory. *Review of Economic Studies* 80, 20 (2012), 925–948.
[10] N. Bulling, J. Dix, and W. Jamroga. 2010. Model Checking Logics of Strategic Ability: Complexity. In *Specification and Verification of Multi-Agent Systems*, M. Dastani, K. Hindriks, and J.-J. Meyer (Eds.). Springer, 125–159.
[11] N. Bulling, V. Goranko, and W. Jamroga. 2015. Logics for Reasoning About Strategic Abilities in Multi-Player Games. In *Models of Strategic Reasoning. Logics, Games, and Communities*, J. van Benthem, S. Ghosh, and R. Verbrugge (Eds.). Lecture Notes in Computer Science, Vol. 8972. Springer, 93–136. https://doi.org/10.1007/978-3-662-48540-8
[12] Jin-Yi Cai. 2012. Lectures in Computational Complexity. (2012).
[13] 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.
[14] K. Chatterjee, T. A. Henzinger, and N. Piterman. 2007. Strategy Logic. In *Proceedings of CONCUR*. 59–73.
[15] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. 2014. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica* 51, 3-4 (2014), 129–163.
[16] Ali Dasdan. 2018. Twelve Simple Algorithms to Compute Fibonacci Numbers. *CoRR* abs/1803.07199 (2018). arXiv:1803.07199
[17] Anindya De, Luca Trevisan, and Madhur Tulsiani. 2009. Non-uniform attacks against one-way functions and PRGs. *Electron. Colloquium Comput. Complex.* TR09-113 (2009). arXiv:TR09-113
[18] L. de Alfaro and T.A. Henzinger. 2000. Concurrent Omega-Regular Games. In *Proceedings of LICS*. 141–154. https://doi.org/10.1109/LICS.2000.855763
[19] W. Diffie and M. Hellman. 1976. New Directions in Cryptography. *IEEE Transactions on Information Theory* 22, 6 (1976), 644–654. https://doi.org/10.1109/TIT.1976.1055638
[20] 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 of Games, Automata, Logics and Formal Verification (GandALF)*. 103–117.
[21] C. Dima and F.L. Tiplea. 2011. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. *CoRR* abs/1102.4225 (2011).
[22] E.A. Emerson. 1990. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science*, J. van Leeuwen (Ed.). Vol. B. Elsevier, 995–1072.
[23] Merrick Furst, James B. Saxe, and Michael Sipser. 1981. Parity, circuits, and the polynomial-time hierarchy. In *22nd Annual Symposium on Foundations of Computer Science (SFCS)*. 260–270. https://doi.org/10.1109/SFCS.1981.35
[24] Oded Goldreich. 1993. A Uniform-Complexity Treatment of Encryption and Zero-Knowledge. *J. Cryptol.* 6, 1 (1993), 21–53. https://doi.org/10.1007/BF02620230
[25] Valentin Goranko, Antti Kuusisto, and Raine Rönnholm. 2021. Game-theoretic semantics for ATL$^+$ with applications to model checking. *Inf. Comput.* 276 (2021), 104554. https://doi.org/10.1016/j.ic.2020.104554
[26] D.P. Guelev and C. Dima. 2012. Epistemic ATL with Perfect Recall, Past and Strategy Contexts. In *Proceedings of Computational Logic in Multi-Agent Systems (CLIMA) (Lecture Notes in Computer Science, Vol. 7486)*. Springer, 77–93. https://doi.org/10.1007/978-3-642-32897-8_7
[27] D.P. Guelev, C. Dima, and C. Enea. 2011. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal of Applied Non-Classical Logics* 21, 1 (2011), 93–131.
[28] J. Hörner and W. Olszewski. 2009. How robust is the Folk Theorem? *The Quarterly Journal of Economics* (2009), 1773–1814.
[29] W. Jamroga and J. Dix. 2006. Model Checking ATL$_{ir}$ is Indeed Δ_2^P-complete. In *Proceedings of EUMAS (CEUR Workshop Proceedings, Vol. 223)*.
[30] Wojciech Jamroga, Vadim Malvone, and Aniello Murano. 2019. Natural Strategic Ability. *Artificial Intelligence* 277 (2019). https://doi.org/10.1016/j.artint.2019.103170
[31] Wojciech Jamroga, Vadim Malvone, and Aniello Murano. 2019. Natural Strategic Ability under Imperfect Information. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*. IFAAMAS, 962–970.
[32] W. Jamroga and W. van der Hoek. 2004. Agents that Know how to Play. *Fundamenta Informaticae* 63, 2–3 (2004), 185–219.
[33] M. Jurdzinski, R. Lazic, and S. Schmitz. 2015. Fixed-Dimensional Energy Games are in Pseudo-Polynomial Time. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, Vol. 2. 260–272.
[34] Jonathan Katz and Yehuda Lindell. 2020. *Introduction to Modern Cryptography, Third Edition*. CRC Press.
[35] Neal Koblitz and Alfred Menezes. 2013. Another look at non-uniformity. *Groups Complex. Cryptol.* 5, 2 (2013), 117–139. https://doi.org/10.1515/gcc-2013-0008
[36] N.R. Kocherlakota. 1998. Money Is Memory. *Journal of Economic Theory* 81, 2 (1998), 232–251. https://doi.org/10.1006/jeth.1997.2357
[37] R. McNaughton. 1966. Testing and generating infinite sequences by a finite automaton. *Information and Control* 9 (1966), 521–530.
[38] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM Transactions on Computational Logic* 15, 4 (2014), 1–42.
[39] F. Mogavero, A. Murano, and M.Y. Vardi. 2010. Reasoning About Strategies. In *Proceedings of FSTTCS*. 133–144.
[40] J. Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann.
[41] A. Pnueli. 1977. The Temporal Logic of Programs. In *Proceedings of FOCS*. 46–57.
[42] 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).
[43] H. G. Rice. 1953. Classes of Recursively Enumerable Sets and Their Decision Problems. *Trans. Amer. Math. Soc.* 74, 2 (1953), 358–366.
[44] A. Rubinstein. 1998. *Modeling bounded rationality*. MIT Press.
[45] P. Y. Schobbens. 2004. Alternating-Time Logic with Imperfect Recall. *Electronic Notes in Theoretical Computer Science* 85, 2 (2004), 82–93.
[46] C.E. Shannon. 1949. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal* 28, 1 (1949), 59–98. https://doi.org/10.1002/j.1538-7305.1949.tb03624.x
[47] Y. Shoham and K. Leyton-Brown. 2009. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.
[48] Stefan Szeider. 2008. *Parameterized SAT*. Springer, 639–640. https://doi.org/10.1007/978-0-387-30162-4_283
[49] W. van der Hoek and M. Wooldridge. 2002. Tractable Multiagent Planning for Epistemic Goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, C. Castelfranchi and W.L. Johnson (Eds.). ACM Press, New York, 1167–1174.
[50] Y. Velner, K. Chatterjee, L. Doyen, Th. A. Henzinger, Al. Rabinovich, and J.-F. Raskin. 2015. The complexity of multi-mean-payoff and multi-energy games. *Information and Computation* 241 (2015), 177–196.
[51] S. Vester. 2013. Alternating-time temporal logic with finite-memory strategies. In *Proceedings of GandALF (EPTCS)*. 194–207. https://doi.org/10.4204/EPTCS.119.17