Reasoning about Natural Strategic Ability

Wojciech Jamroga
Institute of Computer Science,
Polish Academy of Sciences
w.jamroga@ipipan.waw.pl

Vadim Malvone
Universitá degli studi di Napoli
Federico II, Italy
vadim.malvone@unina.it

Aniello Murano
Universitá degli studi di Napoli
Federico II, Italy
murano@na.infn.it

ABSTRACT

In game theory, as well as in the semantics of game logics, a strategy can be represented by any function from states of the game to the agent's actions. That makes sense from the mathematical point of view, but not necessarily in the context of human behavior. This is because humans are quite bad at executing complex plans, and also rather unlikely to come up with such plans in the first place. In this paper, we adopt the view of bounded rationality, and look only at "simple" strategies in specifications of agents' abilities. We formally define what "simple" means, and propose a variant of alternating-time temporal logic that takes only such strategies into account. We also study the model checking problem for the resulting semantics of ability.

1. INTRODUCTION

Logics for strategic reasoning provide powerful tools to reason about multi-agent systems [7, 42, 40, 30, 21, 34]. The logics allow to express properties of agents' behavior and its dynamics, driven by their individual and collective goals. An important factor here is interaction between the agents, which can be cooperative as well as adversarial. Specifications in agent logics can be then used as input to *model checking* [22, 38], which makes it possible to verify the correct behavior of a multi-agent system using recently developed practical automatic tools [33, 19, 20].

A fundamental contribution in this field is *Alternating-Time Temporal Logic (*ATL**)* and its fragment ATL [7]. ATL* formulas are usually interpreted over *concurrent game structures (CGS)* which are labeled state-transition systems that model synchronous interaction among agents. For example, given a CGS modeling a system with k agents and a shared resource, the ATL formula $\langle\!\langle A\rangle\!\rangle \mathsf{F}\mathsf{grant}$ expresses the fact that the set of agents A can ensure that, regardless of the actions of the other agents, an access to the resource will be eventually granted. The specification holds if agents in A have a collective strategy whose every execution path satisfies $\mathsf{F}\mathsf{grant}$. As in game theory, strategies are understood as conditional plans, and play a central role in reasoning about purposeful agents.

Formally, strategies in ATL* (as well as in other logics of strategic reasoning, such as Strategy Logic [21, 34]) are defined as functions from sequences of system states (i.e., possible histories of the game) to actions. A simpler notion of *positional* a.k.a. *memoryless* strategies is formally defined by functions from states to actions.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

That makes sense from a mathematical point of view, and also in case we think of strategic ability of a machine (robot, computer program). We claim, however, that the approach is not very realistic for reasoning about human behavior. This is because humans are very bad at handling combinatorially complex objects. A human strategy should be relatively simple and "intuitive" or "natural" in order for the person to understand it, memorize it, and execute it. This applies even more if the human agent has to come up with the strategy on its own.

In this paper, we adopt the view of bounded rationality, and look only at strategies whose complexity does not exceed a given bound. In this way we put a limit on the resources needed to represent and use the strategy. More precisely, we introduce NatATL*, a logic that extends ATL* by replacing the strategic operator $\langle\!\langle A\rangle\!\rangle\varphi$ with a bounded version $\langle\!\langle A\rangle\!\rangle^{\leq k}\varphi$, where $k \in \mathbb{N}$ denotes the complexity bound. To measure the complexity of strategies, we assume that they are represented by lists of guarded actions. For memoryless strategies, guards are boolean propositional formulas. For strategies with recall, guards are given as regular expressions over boolean propositional formulas. As technical results, we study the problem of model checking NatATL for both memoryless and memoryfull strategies. The complexity ranges from $\mathbf{\Delta_2^P}$ to $\mathbf{\Delta_3^P}$ in the general case, and from $\mathbf{P}$ to $\mathbf{\Delta_2^P}$ for small complexity bounds.

Related Works. ATL* has been the subject of intensive research within multi-agent systems and AI. Works that are closest in spirit to our proposal concern modeling, specification, and reasoning about strategies of bounded agents. The papers that studied explicit representation of strategies are also relevant.

In the former group, [2] investigates strategic properties of agents with bounded memory, while [5, 6, 15, 16] extend temporal and strategic logics to handle agents with bounded resources. Issues related to bounded rationality are also investigated in [10, 28, 25].

The latter category is much richer and includes extensions of ATL* with explicit reasoning about actions and strategies [41, 1, 44, 27], and logics that combine features of temporal and dynamic logic [26, 35]. A variant of STIT logic that enables reasoning about strategies and their performance in the object language [24]. Also, plans in agent-oriented programming are in fact rule-based descriptions of strategies. In particular, reasoning about agent programs using strategic logics was investigated in [12, 3, 4, 23, 45].

None of those works considers directly the subject of this paper, i.e., logic-based reasoning about agents' abilities in scenarios where natural representation and reasonable complexity of strategies is essential.

2. A LOGIC FOR NATURAL ABILITY

In this section we introduce all ingredients to define NatATL, a logic for reasoning about natural strategic ability.

EXAMPLE 1 (MOTIVATING EXAMPLES). *The main application domain that we have in mind is* reasoning about usability. *Consider, e.g., a ticket vending machine at a railway station. Intuitively, it is not enough that a customer has a strategy to successfully buy the right ticket. If the strategy is too complex, most people will be unable to follow it, and the machine will be practically useless.*

Another application area is gaming*, where one could define the game level by the complexity of the smallest winning strategy.*

In both cases, we need to understand what it means for a strategy to be "simple" or "complex", and to relate our definition of strategic ability to this complexity measure.

We begin by presenting the syntax of NatATL. Then, we recall how to model multi-agent systems by means of concurrent game structures. Further, we show how to define natural memoryless strategies based on guarded actions. Finally, we propose the formal semantics of NatATL formulas.

2.1 Syntax

Alternating-time temporal logic (ATL, for short) [7] generalizes branching-time temporal logic CTL* by replacing path quantifiers E, A with *strategic modality* $\langle\langle A\rangle\rangle$. Informally, $\langle\langle A\rangle\rangle\gamma$ reads "there exists a strategy for the coalition A such that, no matter how the other players will act, the formula γ is satisfied. *Natural* ATL (NatATL, for short) is obtained by replacing in ATL the modality $\langle\langle A\rangle\rangle$ with the bounded strategic modality $\langle\langle A\rangle\rangle^{\leq k}$. Intuitively, $\langle\langle A\rangle\rangle^{\leq k}\gamma$ reads as coalition A has a collective strategy of *size less or equal than* k to enforce the property γ. As for ATL, the formulas of NatATL make use of classical temporal operators: "X " ("in the next state"), "G" ("always from now on"), "F" ("now or sometime in the future"), U (strong "until"), and W (weak "until").

Formally, let $\mathcal{A}$ be a finite set of agents and $Prop$ a countable set of atomic propositions. The language of NatATL is defined as follows:

$$\varphi ::= \mathsf{p} \,|\, \neg\varphi \,|\, \varphi \wedge \varphi \,|\, \langle\langle A\rangle\rangle^{\leq k}\mathsf{X}\,\varphi \,|\, \langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{U}\,\varphi \,|\, \langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{W}\,\varphi.$$

where $A \subseteq \mathcal{A}$, $k \in \mathbb{N}$, and $\mathsf{p} \in Prop$. Derived boolean connectives and constants ($\vee, \top, \bot$) are defined as usual. "Sometime" and "always" can be defined as $\mathsf{F}\gamma \equiv \top\,\mathsf{U}\,\gamma$ and $\mathsf{G}\gamma \equiv \gamma\,\mathsf{W}\,\bot$.

Additionally, 1NatATL will denote the fragment of NatATL that admits only formulas consisting of a single strategic modality, followed by a temporal formula over boolean connectives and atomic propositions.

2.2 Concurrent Game Structures

The semantics of NatATL is defined over concurrent game structures[7].

DEFINITION 1 (CGS). *A* concurrent game structure (CGS) *is a tuple $M = \langle \mathcal{A}, St, Act, d, t, Prop, V\rangle$ which includes nonempty finite sets of: agents $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$, states St, actions Act, atomic propositions $Prop$, and a propositional valuation $V : St \to 2^{Prop}$. The function $d : \mathcal{A} \times St \to 2^{Act}$ defines availability of actions. The (deterministic) transition function t assigns a successor state $q' = t(q, \alpha_1, \ldots, \alpha_{|\mathcal{A}|})$ to each state $q \in St$ and any tuple of actions $\alpha_i \in d(a_i, q)$ that can be executed by $\mathcal{A}$ in q.*

In the rest of the paper, we will write $d_a(q)$ instead of $d(a, q)$, and we will denote the set of collective choice of group A at state q by $d_A(q) = \prod_{a_i \in A} d_{a_i}(q)$.

A pointed CGS *is a pair (M, q_0) consisting of a concurrent game structure M and an initial state q_0 in M.*

A *path* $\lambda = q_0q_1q_2\ldots$ in a CGS is an infinite sequence of states such that there is a transition between each q_i, q_{i+1}. $\lambda[i]$ denotes the ith position on λ, $\lambda[i, j]$ the part of λ between positions i and j, and $\lambda[i, \infty]$ the suffix of λ starting with i. We denote with Λ the set of all paths. Similarly, a *history* $h = q_0q_1q_2\ldots q_n$ is a finite sequence of states that can be effected by subsequent transitions. By $last(h) = q_n$ we denote the last element of the sequence. We denote by $H = St^+$ the set of all the histories in the model.

2.3 Strategies and Their Complexity

To properly interpret NatATL formulas, we introduce the concept of *natural strategies* and their *outcomes* over a CGS. Following Schobbens [40], we distinguish between strategies *with* and *without* the recall of the hitherto history of the game. We will use R to refer to the semantics of strategic ability arising for strategies with recall, and r for strategies without recall. In this section, we show how natural strategies without recall can be defined. The other kind of strategies is proposed and studied in Section 4.

We start by defining a *natural memoryless strategy* (or r-strategy) s_a for agent a. The idea is to use a rule-based representation, with a list of *condition-action* rules. The first rule whose condition holds in the current state is selected, and the corresponding action is executed. We formally represent it with *lists of guarded actions*, i.e., sequences of pairs $(\beta(2^{Prop}),\ \alpha)$ such that $\beta(2^{Prop})$ is a boolean combination over possible subsets of $Prop$ and α is an action in $d_a(q)$ for every $q \in St$ such that $q \models \beta(2^{Prop})$, i.e. q satisfies $\beta(2^{Prop})$ w.r.t. the propositional evaluation V. We assume that the last pair on the list is $(\top,\ idle)$, i.e., the last rule is guarded by a condition that will always be satisfied. The set of all natural memoryless strategies is denoted by Σ_a^{r}. By $size(s_a)$, we denote the number of guarded actions in s_a. Moreover, $cond_k(s_a)$ will denote the kth guard (condition) on the list, and $act_k(s_a)$ the corresponding action. Finally, $match(q, s_a)$ is the smallest $n \leq size(s_a)$ such that $q \models cond_n(s_a)$ and $act_n(s_a) \in d_a(q)$. That is, $match(q, s_a)$ matches state q with the first condition in s_a that holds in q, and action available in q.

By $compl(s_a)$, we denote the complexity of the strategy s_a. Intuitively, the complexity of a strategy is understood as the level of sophistication of its representation. Several natural metrics can be used to measure the complexity of a strategy, given its representation from $(\beta(2^{Prop}) \times Act)^+$, e.g.:

Number of used propositions: $compl_{\#}(s_a) = |\{\mathsf{p} \in Prop \mid \mathsf{p} \in dom(s_a)\}|$;

Largest condition: $compl_{\max}(s_a) = \max\{|\phi| \mid (\phi, \alpha) \in s_a\}$;

Total size of the representation: $compl_{\Sigma}(s_a) = \sum_{(\phi,\alpha)\in s_a} |\phi|$

with $|\phi|$ being the number of symbols in ϕ. From now on, we will focus on the last metric for complexity of strategies, which takes into account the total size of all the conditions used in the representation.

EXAMPLE 2. *Consider the following* r*-strategy s:*

1. $(\neg\mathsf{ticket} \wedge \neg\mathsf{selected},\ select)$;
2. $(\neg\mathsf{ticket} \wedge \mathsf{selected},\ pay)$;
3. $(\top,\ idle)$.

If we look at the number of used propositions, we have that $compl_{\#}(s) = |\{\mathsf{ticket}, \mathsf{selected}\}| = 2$. If we consider the largest condition instead, we have $compl_{\max}(s) = 5$. Finally, if we use the total size of the representation, we get $compl_{\Sigma}(s) = 10$.[1]

[1]We leave it as an exercise to the interested reader to construct an equivalent strategy with $compl_{\Sigma}(s) = 8$.

A *collective natural strategy* for agents $A = \{a_1, \dots, a_{|A|}\}$ is a tuple of individual natural strategies $s_A = (s_{a_1}, \dots, s_{a_{|A|}})$. The set of such strategies is denoted by Σ^{r}_A. The "outcome" function $out(q, s_A)$ returns the set of all paths that occur when agents A execute strategy s_A from state q onward. Formally, given a state $q \in St$, a subset of agents A and a collective memoryless strategy s_A, we define:

$$out(q, s_A) = \{\lambda \in \Lambda \mid (\lambda[0] = q) \wedge \forall_{i \geq 0} \exists_{\alpha_1, \dots, \alpha_{|\mathcal{A}|}} .$$
$$(a \in A \Rightarrow \alpha_a = act_{match(\lambda[i], s_a)}(s_a)) \wedge$$
$$(a \notin A \Rightarrow \alpha_a \in d_a(\lambda[i])) \wedge (\lambda[i+1] = t(\lambda[i], \alpha_1, \dots, \alpha_{|\mathcal{A}|}))\}.$$

2.4 Semantics of NatATL

Given a CGS M, a state $q \in St$, a path $\lambda \in \Lambda$, and $k \in \mathbb{N}$, the semantics of NatATL is defined as follows:

$M, q \models_{\mathrm{r}} \mathsf{p}$ iff $\mathsf{p} \in V(q)$, for $\mathsf{p} \in \mathit{Prop}$;

$M, q \models_{\mathrm{r}} \neg\varphi$ iff $M, q \not\models_{\mathrm{r}} \varphi$;

$M, q \models_{\mathrm{r}} \varphi_1 \wedge \varphi_2$ iff $M, q \models_{\mathrm{r}} \varphi_1$ and $M, q \models_{\mathrm{r}} \varphi_2$;

$M, q \models_{\mathrm{r}} \langle\langle A \rangle\rangle^{\leq k} \mathsf{X}\, \varphi$ iff there is a strategy $s_A \in \Sigma^{\mathrm{r}}_A$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[1] \models_{\mathrm{r}} \varphi$;

$M, q \models_{\mathrm{r}} \langle\langle A \rangle\rangle^{\leq k} \mathsf{G} \varphi$ iff there is a strategy $s_A \in \Sigma^{\mathrm{r}}_A$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[i] \models_{\mathrm{r}} \varphi$ for all $i \geq 0$;

$M, q \models_{\mathrm{r}} \langle\langle A \rangle\rangle^{\leq k} \varphi\, \mathsf{U}\, \psi$ iff there is a strategy $s_A \in \Sigma^{\mathrm{r}}_A$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[i] \models_{\mathrm{r}} \psi$ for some $i \geq 0$ and $M, \lambda[j] \models_{\mathrm{r}} \varphi$ for all $0 \leq j < i$.

EXAMPLE 3. *When designing a game, the designer can define the game level by the complexity of the smallest winning strategy for the player. Using* NatATL*, we can say that the level of game G is k iff* $G \models_{\mathrm{r}} \langle\langle a \rangle\rangle^{\leq k} \mathsf{F}\mathsf{win} \wedge \neg \langle\langle a \rangle\rangle^{\leq k-1} \mathsf{F}\mathsf{win}$.

We will refer to the logical system $(\text{NatATL}, \models_{\mathrm{r}})$ as NatATL$_{\mathrm{r}}$, and analogously for 1NatATL$_{\mathrm{r}}$.

3. MODEL CHECKING FOR NATURAL MEMORYLESS STRATEGIES

In this section we show how to solve the model checking problem for NatATL with r-strategies, i.e. NatATL$_{\mathrm{r}}$. We start with the simpler case in which the bound of the strategies is given as a constant and prove that the model checking problem is polynomial in the size of the game structure. Then, we consider the case in which the bound k is a variable and prove that the model checking problem becomes $\mathbf{\Delta_2^P} - complete$. Regarding this latter case, we also investigate the setting in which NatATL$_{\mathrm{r}}$ formulas have only one strategic operator, i.e. 1NatATL$_{\mathrm{r}}$, and show that the model checking problem turns out to be $\mathbf{NP} - complete$. The results and the proofs presented in this section have been inspired by [40, 29].

3.1 Model Checking for Small Strategies

We begin by looking at the model checking of NatATL$_{\mathrm{r}}$ formulas with constant bounds on the strategy modalities. Under this restriction, one can show a polynomial reduction to the model checking problem for CTL formulas. Thus, we obtain the following result.

THEOREM 1. *The model checking problem for* NatATL$_{\mathrm{r}}$ *with fixed k is in* $\mathbf{P}$.

Proof. First, consider the formula $\varphi = \langle\langle A \rangle\rangle^{\leq k} \gamma$, in which $A \subseteq \mathcal{A}$ and γ is a formula over boolean connectives and atomic propositions. By assumption, the collective strategy that we can assign to coalition A, namely s_A, is bounded and precisely it holds that $compl_{\Sigma}(s_A) \leq k$. Thus, we have $O(|Prop|^k)$ possible kinds of guarded actions and so $O((|Prop|^k)^k) = O(|Prop|^{k^2})$ possible lists. Given the collective strategy s_A, we can prune the CGS by removing all edges that disagree with s_A. This operation costs, in the worst case, $O(|t|)$, where t is the transition relation of the input CGS. So far we have solved the strategic operator of the input formula φ and we are left with a structure S that can be seen as a Kripke structure. Now, we can reduce our problem to model checking the CTL formula $\mathsf{A}\gamma$ ("for all paths γ") over S by using the standard model checking algorithm for CTL [22], well-known to have complexity $O(|t| \cdot |\gamma|)$. The total complexity is thus $O(|Prop|^{k^2} \cdot (|t| + (|t| \cdot |\gamma|))) = O(|Prop|^{k^2} \cdot |t| \cdot |\gamma|)$, and hence polynomial in the size of the model.

To conclude the proof, note that if we have a formula with more strategic operators then we can use a classic bottom-up procedure, i.e. we start solving the innermost formula having a strategic operator (as we have done above) and, once this is solved, we update the formula and the structure and continue with the new innermost formula. The procedure ends on dealing with the outermost strategic operator of the input formula. $\square$

3.2 Model Checking: General Case

We now study the complexity for NatATL$_{\mathrm{r}}$ with the bound of the strategic modalities given as variables. We consider two different cases: formulas with a single strategic operator followed by a simple temporal subformula, and formulas with possibly nested strategic operators. For the former case we show an $\mathbf{NP}$ procedure, and by a reduction from SAT that the problem is $\mathbf{NP}$-complete. For the latter case we show a $\mathbf{\Delta_2^P}$ procedure and by a reduction from SNSAT the $\mathbf{\Delta_2^P}$-completeness.

THEOREM 2. *Model checking* 1NatATL$_{\mathrm{r}}$ *is in* $\mathbf{NP}$.

Proof. Consider $\varphi = \langle\langle A \rangle\rangle^{\leq k} \gamma$, in which $A \subseteq \mathcal{A}$ and γ is a formula over boolean connectives and atomic propositions. By assumption, we can use strategies with no a priori bounded size. To overcome this, to construct a collective strategy s_A we use an oracle that returns a collective strategy for A. We can now conclude by using the same reasoning done in the proof of Theorem 1. In particular, since we use an oracle over a polynomial algorithm the overall complexity is $\mathbf{NP}$. $\square$

We continue by showing a matching lower bound by means of a reduction from the well-known SAT problem. We first provide the reduction and then show that it is correct in Theorem 3. In SAT, the main ingredients are a CNF formula $\varphi = C_1 \wedge \dots \wedge C_n$ and m propositional variables from a set $X = \{x_1, \dots, x_m\}$. Each clause C_i can be written as $C_i = x_1^{s(i,1)} \vee \dots \vee x_m^{s(i,m)}$, where $s(i,j) \in \{+, -, 0\}$; x_j^+ denotes a positive occurrence of x_j in C_i, x_j^- denotes an occurrence of $\neg x_j$ in C_i, and x_j^0 indicates that x_j does not occur in C_i. The SAT problem asks if $\exists X.\varphi$, that is, if there is a valuation of $x_1, \dots, x_m$ such that φ holds. We construct the corresponding $CGS\ M_{\varphi}$ as follows. There are two players: verifier $\mathbf{v}$ and refuter $\mathbf{r}$. The state space contains an initial state q_0, a state for each clause C_i in φ, a state for each literal in C_i and the state $q_{\top}$. The set of $Prop$ is $\{\mathsf{C}_1, \dots, \mathsf{C}_n, \mathsf{x}_1, \dots, \mathsf{x}_m, win\}$. Furthermore, we label each state clause/variable with its proposition and $q_{\top}$ with win. The flow of the game is defined as follows. The refuter decides at the beginning of the game which clause C_i will have to be satisfied: it is done by proceeding from the initial

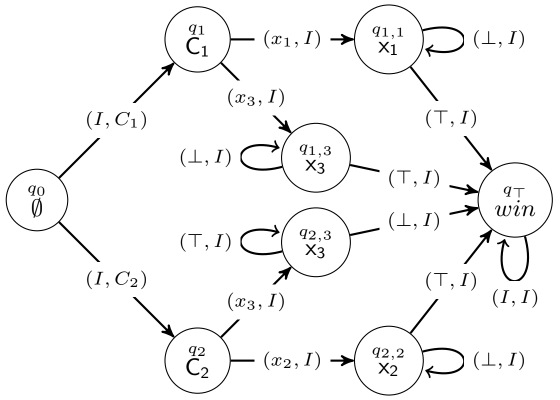

Figure 1: A CGS for checking satisfiability of $\varphi = (x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$. Action I denotes "idle." For simplicity, we omit the states that have no incoming edges.

state q_0 to a clause state q_i. At q_i, verifier decides (by proceeding to a proposition state $q_{i,j}$) which of the literals $x_j^{s(i,j)}$ from C_i will be attempted. Finally, at $q_{i,j}$, verifier attempts to prove C_i by declaring the underlying propositional variable x_j true (action $\top$) or false (action $\bot$). If $\mathbf{v}$ succeeds (i.e., if it executes $\top$ for x_j^+, or executes $\bot$ for x_j^-), then the system proceeds to the winning state $q_\top$. Otherwise, the system stays in $q_{i,j}$. It is important to note that by definitions of $Prop$ and V, we know that $\mathbf{v}$ can use just one action (i.e. truth value) for each variable. This is due to the fact that we use as strategies the guarded actions that are determined directly from the atomic proposition instead from states.

More formally, $M_\varphi = \langle \mathcal{A}, St, Act, d, t, Prop, V\rangle$, where:

- $\mathcal{A} = \{\mathbf{v}, \mathbf{r}\}$,
- $St = \{q_0\} \cup St_{cl} \cup St_{prop} \cup \{q_\top\}$, where $St_{cl} = \{q_1, \ldots, q_n\}$, and $St_{prop} = \{q_{1,1}, \ldots, q_{1,m}, \ldots, q_{n,1}, \ldots, q_{n,m}\}$;
- $Act = \{I, C_1, \ldots, C_n, x_1, \ldots, x_m, \top, \bot\}$,
- $d(\mathbf{v}, q_0) = d(\mathbf{v}, q_\top) = \{I\}$, $d(\mathbf{v}, q_i) = \{x_j \mid x_j$ or $\neg x_j$ is in $C_i\}$, $d(\mathbf{v}, q_{i,j}) = \{\top, \bot\}$; $d(\mathbf{r}, q_0) = \{C_1, \ldots, C_n\}$ and $d(\mathbf{r}, q) = \{I\}$ with $q \in St \setminus \{q_0\}$;
- $t(q_0, I, C_i) = q_i$, $t(q_i, x_j, I) = q_{i,j}$, $t(q_{i,j}, \top, I) = q_\top$ if $s(i,j) = +$, and $q_{i,j}$ otherwise, $t(q_{i,j}, \bot, I) = q_\bot$ if $s(i,j) = -$, and $q_{i,j}$ otherwise;
- $Prop = \{\mathsf{C}_1 \ldots \mathsf{C}_\mathsf{n}, \mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{m}, win\}$;
- $V(q_0) = \emptyset$, $V(q_i) = \mathsf{C}_\mathsf{i}$, $V(q_{i,j}) = \mathsf{x}_\mathsf{j}$, and $V(q_\top) = \mathsf{win}$;

where $1 \leq i \leq n$ and $1 \leq j \leq m$.

As an example, model M_φ for $\varphi = (x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$ is presented in Figure 1.

THEOREM 3. *$SAT(n, m, \varphi)$ iff $M_\varphi, q_0 \models \langle\langle \mathbf{v} \rangle\rangle^{\leq n+m} F\mathsf{win}$*

Proof. ($\Rightarrow$) Firstly, if there is a valuation v that makes φ true, then for every clause C_i one can choose a literal out of C_i that is made true by the valuation. Now, we can construct a strategy for $\mathbf{v}$ such that: (i) for each clause C_i we define a guarded action (C_i, α), where α is the action to go at the state literal that satisfy C_i in accordance with v; and (ii) for each literal x_j we define a guarded action (x_j, α), where α is the action to go in $q_\top$ in accordance with v.
($\Leftarrow$) Conversely, if $M_\varphi, q_0 \models \langle\langle \mathbf{v} \rangle\rangle^{\leq n+m} F\mathsf{win}$, then there is a strategy $s_\mathbf{v}$ such that $q_\top$ is achieved for all paths from $out(q_0, s_\mathbf{v})$. But then the valuation, which assigns propositions $x_1, \ldots, x_m$ with the same values as $s_\mathbf{v}$, satisfies φ. ☐

By Theorem 2 and Theorem 3, the following result holds.

COROLLARY 1. *Model checking* 1NatATL$_\mathrm{r}$ *is* $\mathbf{NP}$*-complete.*

Now, we show how to solve the model checking problem for any formula in NatATL$_\mathrm{r}$.

THEOREM 4. *Model checking* NatATL$_\mathrm{r}$ *is in* $\mathbf{\Delta_2^P}$.

Proof. We make use of a bottom-up procedure based on the one introduced in the proof of Theorem 1. Precisely, take an arbitrary formula φ of NatATL$_\mathrm{r}$ and consider its inner part that is of the kind $\psi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, with γ being a formula over boolean connectives and atomic propositions. Now, apply over ψ the procedure used in the proof of Theorem 2 that we know to be $\mathbf{NP}$. Once ψ is solved, use the same $\mathbf{NP}$ procedure to solve ψ', a formula that contains ψ and a strategic operator, and so on for each strategic operator in φ. This means that we use an oracle over a polynomial procedure for each strategic operator in φ. Summing up, the total complexity to solve a formula in NatATL$_\mathrm{r}$ is $\mathbf{P^{NP}} = \mathbf{\Delta_2^P}$. ☐

We now turn on the lower bound and show a reduction from the SNSAT problem, a well-known $\mathbf{\Delta_2^P}$-hard problem. We first provide the reduction and then prove that it is correct.

DEFINITION 2. *Given a fixed number r and $1 \leq i \leq r$, a SNSAT instance is defined as follows:*

- *r sets of propositional variables $X_i = \{x_{1,i}, \ldots, x_{m,i}\}$;*
- *r propositional variables z_i;*
- *r Boolean formulas φ_i involving only on variables in $X_i \cup \{z_1, \ldots, z_{i-1}\}$;*
- *$z_i \equiv$ there exists an assignment of variables in X_i such that φ_i is true.*

The output of an SNSAT instance is the truth-value of z_r. Note that we can write, by abuse of notation, $z_i \equiv \exists X_i \varphi_i(z_1, \ldots, z_{i-1}, X_i)$. Let n be the maximal number of clauses in any $\varphi_1, \ldots, \varphi_r$ from the given input. Now, each φ_i can be written as:

$$\varphi_i = C_1^i \wedge \ldots \wedge C_n^i, \text{ and}$$

$$C_j^i = x_{1,i}^{s^i(j,1)} \vee \ldots \vee x_{m,i}^{s^i(j,m)} \vee z_1^{s^i(j,m+1)} \vee \ldots z_{i-1}^{s^i(j,m+i-1)}$$

where $1 \leq j \leq n$, $s^i(j,k) \in \{+,-,0\}$ with $1 \leq k \leq m$; as before, $x_{k,i}^+$ denotes a positive occurrence of $x_{k,i}$ in C_j^i, $x_{k,i}^-$ denotes an occurrence of $\neg x_{k,i}$ in C_j^i, and $x_{k,i}^0$ indicates that $x_{k,i}$ does not occur in C_j^i, and $s^i(j,k) \in \{+,-,0\}$ with $m < k < m+i$; defines the sign of z_{k-m} in C_j^i.

Given such an instance of SNSAT, we construct a sequence of concurrent game structures M_i in a similar way to the construction used for the reduction from SAT. That is, clauses and variables $x_{k,i}$ are handled in exactly the same way as before. Moreover, if z_h, with $1 \leq h < i$, occurs as a positive literal in φ_i, we embed M_h in M_i, and add a transition to the initial state q_0^h of M_h. If $\neg z_h$ occurs in φ_i, we do almost the same: the only difference is that we split the transition into two steps, with a state neg_h^i (labeled with a proposition neg) added in between. More formally, $M_i = \langle \mathcal{A}, St^i, Act^i, d^i, t^i, Prop^i, V^i\rangle$, where:

- $\mathcal{A} = \{\mathbf{v}, \mathbf{r}\}$,
- $St^i = \{q_0^i\} \cup St_{cl} \cup St_{prop} \cup St_{neg} \cup \{q_\top\} \cup St^{i-1}$, where $St_{cl} = \{q_1^i, \ldots, q_n^i\}$, $St_{prop} = \{q_{1,1}^i, \ldots, q_{1,m}^i, \ldots, q_{n,1}^i, \ldots, q_{n,m}^i\}$, and $St_{neg} = \{neg_1^i, \ldots, neg_{i-1}^i\}$;

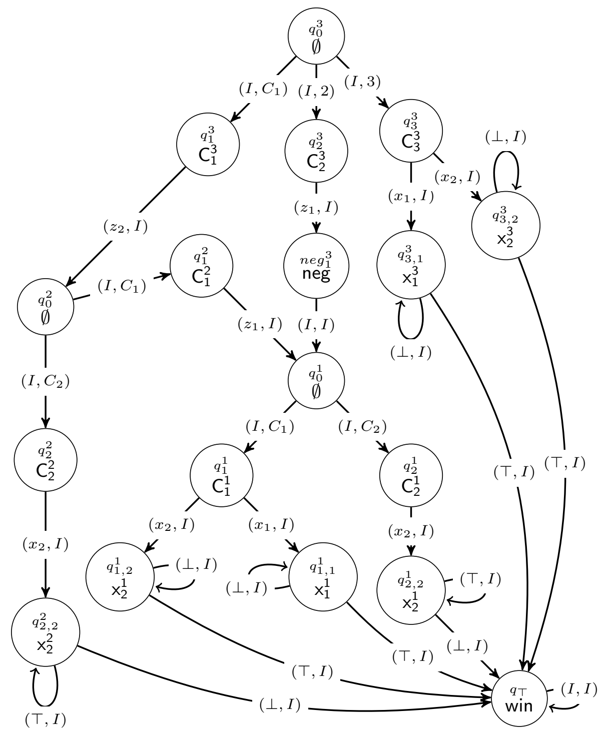

Figure 2: A CGS for $\varphi_3 = z_2 \wedge \neg z_1 \wedge (x_1 \vee x_2)$, $\varphi_2 = z_1 \wedge \neg x_2$, and $\varphi_1 = (x_1 \vee x_2) \wedge \neg x_2$. For simplicity, we omit the states that have no ingoing edges.

- $Act^i = \{I, C_1, \ldots, C_n, x_1, \ldots, x_m, z_1, \ldots, z_r, \top, \bot\}$;
- $d^i(v, q_0^i) = d^i(v, q_\top) = d^i(v, neg_h^i) = \{I\}$, $d^i(v, q_j^i) = \{x_k \mid x_{k,i}$ or $\neg x_{k,i}$ is in $C_j^i\} \cup \{z_h \mid z_h$ or $\neg z_h$ is in $C_j^i\}$, $d^i(v, q_{j,k}^i) = \{\top, \bot\}$; $d^i(\mathbf{r}, q_0^i) = \{C_1, \ldots, C_n\}$ and $d^i(\mathbf{r}, q) = \{I\}$ with $q \in St \setminus \{q_0^i\}$. For $q \in St^{i-1}$, we simply include the function from M_{i-1}: $d^i(a, q) = d^{i-1}(a, q)$;
- $t^i(q_0^i, I, C_j) = q_j^i$, $t^i(q_j^i, x_k, I) = q_{j,k}^i$, $t^i(q_j^i, z_h, I) = q_0^h$ if $s^i(j, h) = +$ and neg_h^i otherwise, $t^i(neg_h^i, I, I) = q_0^h$, $t(q_{j,k}^i, \top, I) = q_\top$ if $s^i(j, k) = +$, and $q_{j,k}^i$ otherwise, $t(q_{j,k}^i, \bot, I) = q_\top$ if $s^i(j, k) = -$, and $q_{i,j}$ otherwise. For $q \in St^{i-1}$, we include the transition function from M_{i-1}: $t^i(q, \alpha_1, \alpha_2) = t^{i-1}(q, \alpha_1, \alpha_2)$;
- $Prop^i = \{\mathsf{C}_1^i, \ldots, \mathsf{C}_n^i, \mathsf{x}_1^i, \ldots, \mathsf{x}_m^i, \mathsf{win}, \mathsf{neg}\}$;
- $V(q_0^i) = \emptyset$, $V(q_j^i) = \mathsf{C}_j^i$, $V(q_{j,k}^i) = \mathsf{x}_k^i$, $V(neg_h^i) = \mathsf{neg}$ and $V(q_\top) = \mathsf{win}$.

where $1 \le i \le r$, $1 \le j \le n$, $1 \le k \le m$, and $1 \le h < i$.

As an example, model M_3 is presented in Figure 2.

To prove the hardness, we consider the following sequence of formulas.

$$\phi_1 = \langle\langle v \rangle\rangle^{\le n+m} (\neg \mathsf{neg})\, \mathsf{U}\, (\mathsf{win}),$$
$$\vdots$$
$$\phi_i = \langle\langle v \rangle\rangle^{\le n+m} (\neg \mathsf{neg})\, \mathsf{U}\, (\mathsf{win} \vee (\mathsf{neg} \wedge \langle\langle \emptyset \rangle\rangle^{\le 0} \mathsf{X}\, \neg \phi_{i-1})).[2]$$

[2]Note that $\langle\langle \emptyset \rangle\rangle^{\le 0}$ is equivalent to the CTL path quantifier A ("for all paths"). To see this, observe that the empty coalition $\emptyset$ has just one strategy – the empty strategy, which is of size 0 – and the strategy enforces γ iff γ holds on all paths in the system, starting from the current state.

Before we prove the hardness, we state an important lemma. It says that overlong formulas ϕ_i do not introduce new properties of model M_l, with $1 \le l \le i \le r$. More precisely, a formula ϕ_i that includes more nestings than model M_l can be as well reduced to ϕ_{i-1} when model checked in M_l, q_0^l.

LEMMA 1. *$\forall 1 \le l \le i \le r$: $M_l, q_0^l \models \phi_i$ iff $M_l, q_0^l \models \phi_{i-1}$.*

The proof of the lemma is a straightforward adaptation of [29, Lemma 5].

THEOREM 5. *$\forall 1 \le i \le r$: z_i is true iff $M_i, q_0^i \models \phi_i$*

Proof. Induction on i:
(i) For $i = 1$, we use the proof of Theorem 3.
(ii) For $i > 1$, we prove both directions.
($\Rightarrow$) Firstly, if z_i is true then there is a valuation v of X_i that makes φ_i true. We construct s_v as in the proof of Theorem 3. In case that some $x_{k,i}^s$ has been chosen in clause C_j^i then we define the guarded action $(\mathsf{C}_j^i,\ x_k)$ and we are done. In case that some z_h^- has been chosen in clause C_j^i, where $h < i$, we have (by induction) that $M_h, q_0^h \models \neg\phi_h$. By Lemma 1, also $M_h, q_0^h \models \neg\phi_i$, and hence $M_i, q_0^h \models \neg\phi_i$. So we can make the same choice (i.e., we define the guarded action $(\mathsf{C}_j^i,\ z_h)$) in s_v, and this will lead to state neg_h^i, in which it holds that $neg \wedge A\mathsf{X}\, \neg\phi_i$. In case that some z_h^+ has been chosen in clause C_j^i, we have that $M_h, q_0^h \models \phi_h$. By Lemma 1, also $M_h, q_0^h \models \phi_i$. That is, there is a strategy s_v' in M_h such that $(\neg \mathsf{neg})\, \mathsf{U}\, (\mathsf{win} \vee (\mathsf{neg} \wedge A\mathsf{X}\, \neg\phi_{i-1}))$ holds for all paths from $out(q_0^h, s_v')$. Then, we can merge s_v' into s_v.
($\Leftarrow$) Conversely, if $M_i, q_0^i \models \phi_i$, then there is a strategy s_v that enforces $(\neg \mathsf{neg})\, \mathsf{U}\, (\mathsf{win} \vee (\mathsf{neg} \wedge A\mathsf{X}\, \neg\phi_{i-1}))$. First, we consider the clause C_j^i with guarded action $(\mathsf{C}_j^i,\ x_k)$, i.e. for which a propositional state is chosen by s_v. The strategy defines a valuation for X_i that satisfies these clauses. For the other clauses, i.e. there is a guarded action $(\mathsf{C}_j^i,\ z_h)$, we have two possibilities:

- s_v chooses q_0^h in the state corresponding to C_j^i. Neither win nor neg have been encountered on this path yet, so we can take s_v to demonstrate that $M_i, q_0^h \models \phi_i$, and hence $M_h, q_0^h \models \phi_i$. By Lemma 1, also $M_h, q_0^h \models \phi_h$. By induction, z_h must be true, and hence clause C_j^i is satisfied.
- s_v chooses neg_h^i in the state corresponding to C_j^i. Then, it must be that $M_i, neg_h^i \models A\mathsf{X}\, \neg\phi_{i-1}$, and hence $M_h, q_0^h \models \neg\phi_{i-1}$. By Lemma 1, also $M_h, q_0^h \models \neg\phi_h$. By induction, z_h must be false, and hence clause C_j^i (containing $\neg z_h$) is also satisfied. □

By Theorem 4 and Theorem 5, the following result holds.

COROLLARY 2. *Model checking* NatATL$_\mathrm{r}$ *is* $\mathbf{\Delta_2^P}$*-complete.*

4. A LOGIC FOR NATURAL STRATEGIC ABILITY OF AGENTS WITH MEMORY

Agents with memory can base their decisions on the history of the game, that has occurred so far. We represent conditions on histories by regular expression over boolean propositional formulas.

4.1 Natural Recall

Let $Reg(L)$ be the set of regular expressions over the language L (with the standard constructors $\cdot, \cup, *$ representing concatenation, nondeterministic choice, and finite iteration). A *natural strategy with recall* (or R-strategy) s_a for agent a is a sequence of appropriate pairs from $Reg(\beta(2^{Prop})) \times Act$. That is, it consists of pairs $(r,\ \alpha)$ where r is a regular expression over $\beta(2^{Prop})$ and α is an action available in $last(h)$, i.e. $\alpha \in d_a(last(h))$, for all histories $h \in H$ consistent with r. Formally, given a regular expression r and the language $L(r)$ on words generated by r, a history $h = q_0 \dots q_n$ is consistent with r iff $\exists\, b \in L(r)$ such that $|h| = |b|$ and $\forall_{0 \le i \le n}\ h[i] \models b[i]$. Similarly to r-strategies, the last pair on the list is assumed to be simply $(\top^*,\ idle)$. The set of such strategies is denoted by Σ_a^{R}. Finally, $match(\lambda[0,i], s_a)$ is the smallest $n \le size(s_a)$ such that $\forall_{0 \le j \le i} \lambda[j] \models cond_n(s_a)[j]$ and $act_n(s_a) \in d_a(\lambda[i])$. A *collective natural strategy* for agents $A = \{a_1, \dots, a_{|A|}\}$ is a tuple of individual natural strategies $s_A = (s_{a_1}, \dots, s_{a_{|A|}})$. The set of such strategies is denoted by Σ_A^{R}. Again, $out(q, s_A)$ returns the set of all paths of strategy s_A. For strategies with recall, we simply replace "$match(\lambda[i], s_a)$" with "$match(\lambda[0,i], s_a)$" in the definition from Section 2.3.

We extend the metrics to strategies with recall and collective strategies with recall in the straightforward way.

EXAMPLE 4. *Consider the following* R*-strategy s:*

1. $(\mathsf{safe}^*,\ digGold)$;
2. $(\mathsf{safe}^* \cdot (\neg\mathsf{safe} \wedge \mathsf{haveGun}),\ shoot)$;
3. $(\mathsf{safe}^* \cdot (\neg\mathsf{safe} \wedge \neg\mathsf{haveGun}),\ run)$;
4. $(\top^* \cdot (\neg\mathsf{safe}) \cdot (\neg\mathsf{safe}),\ hide)$;
5. $(\top^*,\ idle)$.

(1) represents the guarded action in which safe *has held in all the states of the history. In that case, the agent should quietly dig for gold. Otherwise, (2) or (3) is used for each history in which* safe *held for all states but the last. Then, the agent should run away or shoot back depending on whether she has a gun. If it doesn't work (item (3)), the agent should hide. Otherwise (item (4)), she waits and does nothing. For the complexity, we have that $compl_{\#}(s) = 2$, $compl_{\max}(s) = 8$, and $compl_{\Sigma}(s) = 27$.*

REMARK 1. *Note that natural strategies with recall are by definition finite. Thus, they do not exactly correspond to the notion of* perfect recall *where an agent may specify different choices for each of the infinitely many finite histories of the game. In this sense, our representations are similar to* finite memory strategies *from [43]. We will look closer at the connection in Section 4.4.*

4.2 NatATL for Strategies with Recall

Now it is easy to define the semantics of natural strategic ability for agents with recall. Formally, we construct the semantic relation $\models_{\mathrm{R}}$ by replacing "$\models_{\mathrm{r}}$" with "$\models_{\mathrm{R}}$" and Σ_A^{r} with Σ_A^{R} in the clauses from Section 2.4.

We will refer to the logical system $(\mathrm{NatATL}, \models_{\mathrm{R}})$ as $\mathrm{NatATL}_{\mathrm{R}}$.

4.3 Relation to Natural Memoryless Strategies

It is well known that the semantics of ATL based on memoryless and perfect recall strategies coincide (under perfect information). This follows from the correctness of the model checking algorithm in [7], cf. also [40]. Precisely, there is a strategy with recall to enforce a given temporal property γ iff there is a memoryless strategy to enforce γ. We now prove that the same does not hold in NatATL.

THEOREM 6. *The following results hold in* NatATL*:*

1. *For all M, q, and all formulas $\varphi = \langle\!\langle A \rangle\!\rangle^{\le k}\gamma$, it holds that $M, q \models_{\mathrm{r}} \varphi$ implies $M, q \models_{\mathrm{R}} \varphi$.*
2. *There exist M, q, and a formula $\varphi = \langle\!\langle A \rangle\!\rangle^{\le k}\gamma$, such that $M, q \models_{\mathrm{R}} \varphi$ and not $M, q \models_{\mathrm{r}} \varphi$.*

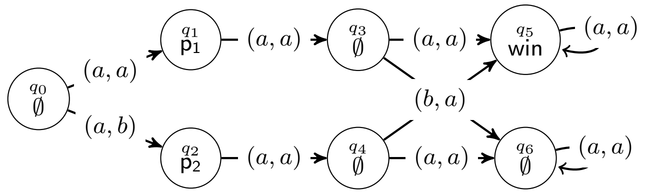

Figure 3: A counterexample for Theorem 6.

Proof. (1) Firstly, given an r-strategy s it is possible to construct an R-strategy s' that has the same behavior of s. In fact, for each guarded action $(\theta,\ \alpha)$ of s with $\theta \in \beta(2^{Prop})$ and $\alpha \in Act$ we can write a guarded action $(r,\ \alpha)$ in s' such that $r = (\top^*) \cdot \theta$.
(2) Consider, the $CGS\ M$ in Figure 3, where there are two players 1 and 2. Each transition is labeled with a couple of actions (α, β), where α is an action of 1 and β is an action of 2. We show that the formula $\varphi = \langle\!\langle 1 \rangle\!\rangle^{\le k} F\mathsf{win}$ is true by using a natural strategy with recall and is false for each possible natural memoryless strategy. The following strategy with recall s satisfies φ:

- $(\top \cdot p_1 \cdot \top,\ a)$;
- $(\top \cdot p_2 \cdot \top,\ b)$;
- $(\top^*,\ a)$.

To be convinced, first recall that natural memoryless strategies are defined only over atomic propositions, so, if there are states with the same set of atomic propositions, then there exists at most one guarded action and then at most one possible action in these states. Therefore, in the model in Figure 3 with a natural memoryless strategy it is impossible to define two different behaviors in the states q_3 and q_4, and for this reason player 1 has no memoryless strategies to reach a state labeled with win. $\square$

Note that the proof of (2) does not use the bound k to construct the counterexample. Thus, it is not only the case that a strategy with recall may inflate beyond the given bound when being transformed to memoryless; it may even be the case that an equivalent natural memoryless strategy does not exist! This is because in NatATL choices in strategies are based on conditions whose granularity depends on the available Boolean propositions. In contrast, the semantics of ATL defines memoryless strategies as functions from states to actions, which allows for arbitrary granularity.

To remove the limit of natural memoryless strategies, we define a subclass of models named *fully distinguishing models*. The idea behind this kind of models is the one used in [7, 32] to define the distinguishing models. The formal definition follows.

DEFINITION 3. *Given a CGS M, we say that M is a* fully distinguishing model *iff, for all $S \subseteq St$, there exists $\mathsf{p} \in Prop$ such that:*

- $\forall s \in S,\ M, s \models \mathsf{p}$, *and*
- $\forall s \notin S,\ M, s \not\models \mathsf{p}$

THEOREM 7. *Given a fully distinguishing model M, a state q, a subset of agents A, and a formula $\varphi = \langle\!\langle A \rangle\!\rangle^{\le k}\gamma$, it holds that: $M, q \models_{\mathrm{r}} \varphi$ iff $M, q \models_{\mathrm{R}} \varphi$.*

Proof. ($\Rightarrow$) For this direction we use the proof of Theorem 6(1).
($\Leftarrow$) Assume now that $M, q \models_{\mathrm{R}} \varphi$. By definition, there is a strategy $s_A \in \Sigma_A^{\mathrm{R}}$ such that $compl(s_A) \le k$, and for each path

$\lambda \in out(q, s_A)$, we have $M, \lambda \models_{\mathrm{R}} \gamma$. From s_A, let us construct a memoryless strategy $s'_A \in \Sigma^{\mathrm{r}}_A$ such that the following facts hold: (i) $\forall \lambda \in out(q, s'_A)$, we have $M, \lambda \models_{\mathrm{r}} \gamma$ and (ii) $compl(s'_A) \leq compl(s_A)$. We start at the state q. We know that M is a fully distinguishing model, so the state q is distinguishable with respect to the other states of M. Consider for simplicity that the only atomic proposition that is true in q is q. We fix $s'_A(\mathsf{q}) = s_A(\mathsf{q})$, where $s_A(\mathsf{q})$ represents the action in the strategy with recall s_A for the regular expression q that is just an atomic proposition. Consider now the successors of q consistent with $s_A(\mathsf{q})$. $\forall q' \in out(q, s_A(\mathsf{q}))$ we take the atomic proposition q' that is true just in q' and fix $s'_A(\mathsf{q}') = s_A(\mathsf{q} \cdot \mathsf{q}')$, where $\mathsf{q} \cdot \mathsf{q}'$ is the regular expression that is composed by the atomic propositions q and q' that are only true in q and q', respectively. We repeat this procedure until we get to a fixpoint, i.e. all states are covered, except possibly for some states that are unreachable when we execute s_A. By the definition, we also know that these states satisfy the guarded action $(\top,\ idle)$. To conclude the proof, we just need to show that (i) and (ii) hold. Item (i) can be proved by induction. For the lack of space, we omit the details. Item (ii) follows by the construction of s'_A. In fact, we construct s'_A from s_A, that is for each guarded action $(\mathsf{q},\ \alpha)$ of s'_A there is a guarded action $(r,\ \alpha)$ of s_A, where $r = r_0 \cdot \ldots \cdot r_n$ and $r_n = \mathsf{q}$, then $compl(s'_A) \leq compl(s_A)$. □

4.4 Correspondence to DFST

In [43], another useful representation of finite-memory strategies has been introduced by means of *deterministic finite-state transducers (DFST's)*. A DFST is a tuple $(V, v_0, In, Out, F_{in}, F_{out})$, where V is a finite non-empty set of states, v_0 is the initial state, In is the input alphabet, Out is the output alphabet, $F_{in} : V \times In \to V$ is the transition function and $F_{out} : V \times In \to Out$ is the output function. The set V represents the possible values of the internal memory of the strategy. The initial state corresponds to the initial memory value. The input and output symbols are the states and the actions of the game, respectively. In each round of the game the DFST reads a state of the game. Then it updates its memory based on the current memory value and the input state and performs an action. Formally, a strategy $s : H \to Act$ is a finite-memory strategy if there exists a DFST such that for all $h \in H$: $s(h) = F_{out}(G(v_0, h[0, |h| - 1]), last(h))$, where G is defined recursively by $G(v, \rho) = In(v, \rho_0)$ for any state v and any history ρ with $|\rho| = 0$ and $G(v, \rho) = In(G(v, \rho[0, |\rho| - 1]), last(\rho))$ for any state v and any history ρ with $|\rho| > 0$. Intuitively G is the function that repeatedly applies the transition function In on a sequence of inputs to calculate the state after a given history.

In our setting, given a strategy $s : H \to Act$, it is possible to rewrite s in a natural strategy with recall s' if for all $h \in H$ we have that $s(h) = \alpha$, where $(r,\ \alpha)$ is the first pair in s' in which h is consistent with r.

We now compare natural strategies with recall vs. finite-memory strategies, the way they are defined in [43]. In particular, we prove the following result.

THEOREM 8. *Natural strategies with recall are equally expressive and more succinct than finite-memory strategies represented by DFST.*

Proof. To prove that the two representations *are equally expressive*, we show that from a natural strategy with recall s we can construct an equivalent DFST D (possibly with an exponential blowup) and from a DFST D we can construct an equivalent natural strategy with recall s (by a polynomial construction).
($\Rightarrow$) Firstly, given s we show how to construct D. Recall that s is a sequence of pairs $(r,\ \alpha)$. We know that with a sequence of regular expressions it is possible to construct, in polynomial time, a non-deterministic finite automaton N. By means of a classic *powerset construction*, given N of size n, we can construct a deterministic finite automaton A with size 2^n [39], i.e. in an exponential blowup.
($\Leftarrow$) Conversely, given D we now sketch how to construct an equivalent natural strategy s. Basically, for each edge e in D, i.e. for each $(v, q) \in dom(F_{In})$, we construct a guarded action $(r,\ \alpha)$, where $r = r_0 \cdot \ldots \cdot r_l$, in which $r_0 \cdot \ldots \cdot r_{l-1}$ is derived from the actual state of D to the initial state of D (w.r.t backward) and $r_l = q$, and $\alpha = F_{Out}(v, q)$.

Finally, the fact that natural strategies with recall *are more succinct* is a consequence of the fact that there a minimal NFA that are exponentially smaller than the minimal equivalent DFA [39]. □

5. MODEL CHECKING FOR NATURAL STRATEGIES WITH RECALL

In this section we show how to solve the model checking problem for NatATL with R-strategies, i.e. $\mathrm{NatATL_R}$. We consider both the cases in which the bound of the strategies is a constant or a variable.

5.1 Model Checking for Small Strategies

When the bound of the strategies is fixed, we can reduce our problem to the model checking for CTL. This leads to the following result.

THEOREM 9. *The model checking problem for* $\mathrm{NatATL_R}$ *with fixed k is in* $\mathbf{\Delta_2^P}$.

Proof. Assume for the moment that we have a $\mathrm{NatATL_R}$ formula $\varphi = \langle\langle A \rangle\rangle^{\leq k} \gamma$, where $A \subseteq \mathcal{A}$ and γ is a formula over boolean connectives and atomic propositions. As for the solution in $\mathrm{NatATL_r}$, we know that the collective strategy we can assign to A, namely s_A, is bounded and, precisely, we have that $compl_{\Sigma}(s_A) \leq k$. The main difference between r-strategies and R-strategies regards the underlying domains, i.e., we move from boolean propositional formulas to regular expressions over boolean propositional formulas (both over atomic propositions). Recall that, regular expressions are a combination of atomic propositions ($Prop$), boolean connectives ($Bool$), and standard constructors (Con). Thus, in this case, we have $(|Prop + Bool + Con|)^k$ possible different guarded actions and $(|Prop + Bool + Con|^k)^k = |Prop + Bool + Con|^{k^2}$ possible lists. Given s_A, we cannot prune M since we have an R-strategy. Let us consider now the unwinding of M and remove all edges that are not in accordance with s_A. It is important to observe that the unwinding of a model can be infinite and thus we need to consider a bounded unwinding. A possibility would be to consider the tree unwinding with depth $|St| + 1$ as we are sure that after this bound there is a loop. Unfortunately, this is a too big upper-bound. Indeed, checking all paths of the unwinding, in the worst case (i.e. each state is connected with all states of the model), requires $|St|^{|St|}$ steps, that is exponential on the number of states. To avoid this exponential blow-up we use a guessing oracle. The algorithm that solves the model checking problem for $\mathrm{NatATL_R}$ with fixed k is depicted in Figure 4. The $mCheck^k_{\mathrm{NatATL_R}}$ algorithm uses the oracle depicted in Figure 5. It guesses a history h of length $|St| + 1$ that satisfies the CTL formula $\neg \mathsf{A}\gamma$. If such a history exists then the oracle returns true and $mCheck^k_{\mathrm{NatATL_R}}$ returns false because the original formula wants γ to hold. Conversely, if the oracle returns false and $mCheck^k_{\mathrm{NatATL_R}}$ returns true then $M, q \models_{\mathrm{R}} \varphi$. For the complexity, we use an oracle over a polynomial procedure. So, we have that the total complexity is

```
Algorithm $mCheck^k_{\text{NatATL}_{\text{R}}}$(M, q, $\varphi$):
for every $s_A$ with $compl(s_A) \leq k$ do
$t = Oracle(M, q, \varphi, s_A)$
return $(\neg t)$
```

Figure 4: Model checking algorithm for NatATL$_{\text{R}}$ with fixed k

```
Algorithm Oracle(M, q, $\varphi$, $s_A$):
Guess $h \in H^{|St+1|}(q)$
if $h$ is inconsistent with $s_A$
  return false
else
  return $mCheck_{\text{CTL}}(h, h[0], \neg \mathsf{A}\gamma)$
```

Figure 5: Oracle

$\mathbf{P}^{\mathbf{NP}} = \mathbf{\Delta_2^P}$. To conclude the proof, let us drop the initial limitations on the formula. If the formula has more than one strategic operator, then we proceed as in the proof of Theorem 1 and so we use a bottom-up procedure, i.e. we first solve the formula with the inner most strategic operator, then we update the formula and repeat the procedure, until we reach the outermost formula. This requires to use a further oracle over a polynomial procedure that works over a polynomial procedure itself. Hence, we have that the overall complexity is $\mathbf{P}^{\mathbf{P}^{\mathbf{NP}}} = \mathbf{P}^{\mathbf{NP}} = \mathbf{\Delta_2^P}$. $\square$

5.2 Model Checking: General Case

We now study the complexity for NatATL$_{\text{R}}$ in case the bound over the strategies is not fixed. In particular, we study separately the cases in which the formula under exam has one or more nested strategic operators. For the former we show a $\mathbf{\Sigma_2^P}$ procedure and, for the latter, a $\mathbf{\Delta_3^P}$ one. As for the memoryless case, the proofs in this section have been inspired by [40, 29].

THEOREM 10. *Model checking* 1NatATL$_{\text{R}}$ *with variable k is in* $\mathbf{\Sigma_2^P}$.

Proof. Consider the formula $\langle\langle A\rangle\rangle^{\leq k}\gamma$, where $A = \{a\}$ and γ is a formula over boolean connectives and atomic propositions. By assumption, the bound of the strategy is not fixed. For this reason, to construct a strategy s_a we use an $\mathbf{NP}$ oracle that constructs a strategy for a. We report in Figure 6 the related $mCheck_{\text{NatATL}_{\text{R}}}$ algorithm. Regarding the complexity, since we use an oracle over a non-deterministic algorithm we have that checking the model checking problem is $\mathbf{NP}^{\mathbf{NP}} = \mathbf{\Sigma_2^P}$. $\square$

THEOREM 11. *Model checking* NatATL$_{\text{R}}$ *with variable k is in* $\mathbf{\Delta_3^P}$.

Proof. We can use a bottom-up procedure similarly to the one we have used in the proof of Theorem 1 for NatATL$_{\text{r}}$, by looping the construction in Theorem 10. In this case, we use an oracle over a non-deterministic procedure over a polynomial procedure, so we obtain that the overall complexity to solve the addressed problem is $\mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}} = \mathbf{\Delta_3^P}$. $\square$

```
Algorithm $mCheck_{\text{NatATL}_{\text{R}}}$(M, q, $\varphi$, k):
Guess $s_A$ with $compl(s_A) \leq k$
$t =$ Oracle($M, q, \varphi, s_A$)
return $(\neg t)$
```

Figure 6: Model checking NatATL$_{\text{R}}$ with variable k

	memoryless	finite recall
ATL	$\mathbf{P}$-complete	$\mathbf{P}$-complete
1NatATL, fixed k	in $\mathbf{P}$	in $\mathbf{\Delta_2^P}$
NatATL, fixed k	in $\mathbf{P}$	in $\mathbf{\Delta_2^P}$
1NatATL, variable k	$\mathbf{NP}$-complete	in $\mathbf{\Sigma_2^P}$
NatATL, variable k	$\mathbf{\Delta_2^P}$-complete	in $\mathbf{\Delta_3^P}$

Figure 7: Summary of model checking complexity results

6. SUMMARY AND FUTURE WORK

In this paper, we propose an alternative take on strategic reasoning, where agents can handle only relatively simple strategies. We use a natural representation of strategies by lists of guarded actions, and assume that only strategies up to size k can be employed as witnesses to formula $\langle\langle A\rangle\rangle^{\leq k}\gamma$. In terms of technical results, we show that model-checking for NatATL with memoryless strategies is in $\mathbf{P}$ when k is fixed, and $\mathbf{\Delta_2^P}$-complete when k is a parameter of the problem. For strategies with recall, the problem is in $\mathbf{\Delta_2^P}$ when k is fixed, and in $\mathbf{\Delta_3^P}$ in the general case, cf. the summary presented in Figure 7. Clearly, reasoning about *simple* natural memoryless strategies is no more difficult than about arbitrary ATL strategies (and in practice we expect it to be actually easier). On the other hand, verification of natural strategies with recall seems distinctly harder. It would be interesting to look for conditions under which the latter kind of strategies can be synthesized in polynomial time.

We also prove an important property that sets NatATL apart from standard ATL: in NatATL, the memoryless and memoryfull semantics do not coincide. Moreover, we show that our representation of memoryfull strategies is equally expressive, and strictly more succinct, than deterministic finite-state transducers from [43]. As a consequence, model checking natural strategic ability in unbounded strategies with recall is undecidable.

In the future, we plan to extend the framework to natural strategies with imperfect information. We would also like to extend our results to the broader language of NatATL*, and refine them in terms of parameterized complexity. Another interesting path concerns a graded version of the logic with counting how many successful natural strategies are available. We also intend to look at other natural expressions of strategies, including a survey of psychological studies suggesting how people plan and execute their long-term behaviors. Finally, a more complete account of bounded rationality may be obtained by combining bounds on conceptual complexity of strategies (in the spirit of our work here) with their temporal complexity via timing constraints in the vein of [14, 9].

Acknowledgements. We thank the anonymous reviewers for their useful comments. Wojciech Jamroga acknowledges the support of the National Centre for Research and Development (NCBR), Poland, under the PolLux project VoteVerif (POLLUX-IV/1/2016). Vadim Malvone and Aniello Murano acknowledge the support of the GNCS 2017 project "Logiche e Automi per il Model Checking Intervallare".

REFERENCES

[1] T. Ågotnes. Action and knowledge in alternating-time temporal logic. *Synthese*, 149(2):377–409, 2006. Section on Knowledge, Rationality and Action.

[2] T. Ågotnes and D. Walther. A logic of strategic ability under bounded memory. *Journal of Logic, Language and Information*, 18(1):55–77, 2009.

[3] N. Alechina, M. Dastani, B. Logan, and J.-J. C. Meyer. A logic of agent programs. In *AAAI*, pages 795–800, 2007.

[4] N. Alechina, B. Logan, M. Dastani, and J.-J. C. Meyer. Reasoning about agent execution strategies. In *AAMAS*, pages 1455–1458, 2008.

[5] N. Alechina, B. Logan, N. Nga, and A. Rakib. A logic for coalitions with bounded resources. In *IJCAI*, pages 659–664, 2009.

[6] N. Alechina, B. Logan, H. Nguyen, and A. Rakib. Resource-bounded alternating-time temporal logic. In *AAMAS*, pages 481–488, 2010.

[7] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[8] B. Aminof, A. Murano, S. Rubin, and F. Zuleger. Prompt alternating-time epistemic logics. In *KR*, pages 258–267, 2016.

[9] E. Andre, W. Jamroga, M. Knapik, W. Penczek, and L. Petrucci. Timed atl: Forget memory, just count. In *AAMAS*, 2017. To appear.

[10] M. Barlo, G. Carmona, and H. Sabourian. Bounded memory with finite action spaces. *Sabanci University, Universidade Nova de Lisboa and University of Cambridge*, 2008.

[11] A. Bianco, F. Mogavero, and A. Murano. Graded computation tree logic. *ACM Trans. Comput. Log.*, 13(3):25, 2012.

[12] R. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, 2006.

[13] T. Brihaye, A. D. Costa, F. Laroussinie, and N. Markey. Atl with strategy contexts and bounded memory. In *LFCS*, LNCS 5407, pages 92–106, 2009.

[14] T. Brihaye, F. Laroussinie, N. Markey, and G. Oreiby. Timed concurrent game structures. In *CONCUR*, LNCS 4703, pages 445–459, 2007.

[15] N. Bulling and B. Farwer. Expressing properties of resource-bounded systems: The logics RTL* and RTL. In *CLIMA*, LNCS 6214, pages 22–45, 2010.

[16] N. Bulling and B. Farwer. On the (un-)decidability of model checking resource-bounded agents. In *ECAI*, FAIA 215, pages 567–572, 2010.

[17] N. Bulling and W. Jamroga. Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. *Autonomous agents and multi-agent systems*, 28(3):474–518, 2014.

[18] S. Busard, C. Pecheur, H. Qu, and F. Raimondi. Reasoning about memoryless strategies under partial observability and unconditional fairness constraints. *Information and Computation*, 242:128–156, 2015.

[19] P. Čermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications. In *CAV*, LNCS 8559, pages 524–531, 2014.

[20] P. Čermák, A. Lomuscio, and A. Murano. Verifying and Synthesising Multi-Agent Systems against One-Goal Strategy Logic Specifications. In *AAAI*, pages 2038–2044, 2015.

[21] K. Chatterjee, T. Henzinger, and N. Piterman. Strategy Logic. *Information and Computation*, 208(6):677–693, 2010.

[22] E. Clarke and E. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *LP*, LNCS 131, pages 52–71, 1981.

[23] M. Dastani and W. Jamroga. Reasoning about strategies of multi-agent programs. In *AAMAS*, pages 625–632, 2010.

[24] H. Duijf and J. M. Broersen. Representing strategies. In *SR*, EPTCS 218, pages 15–26, 2016.

[25] A. Gupta, S. Schewe, and D. Wojtczak. Making the best of limited memory in multi-player discounted sum games. *arXiv preprint arXiv:1410.4154*, 2014.

[26] D. Harel and D. Kozen. Process logic: Expressiveness, decidability, completeness. *Journal of Computer and System Sciences*, 25(2):144–170, 1982.

[27] A. Herzig, E. Lorini, F. Maffre, and D. Walther. Alternating-time temporal logic with explicit programs. In *LAMAS*, 2014.

[28] J. Hörner and W. Olszewski. How robust is the folk theorem? *The Quarterly Journal of Economics*, pages 1773–1814, 2009.

[29] W. Jamroga and J. Dix. Model checking ATL_{ir} is indeed Δ_2^P-complete. In *EUMAS*, CEUR 223, 2006.

[30] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004.

[31] P. Kaźmierczak, T. Ågotnes, and W. Jamroga. Multi-agency is coordination and (limited) communication. In *PRIMA*, LNCS 8861, pages 91–106, 2014.

[32] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *arXiv preprint arXiv:0804.2435*, 2008.

[33] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *CAV*, LNCS 5643, pages 682–688, 2009.

[34] F. Mogavero, A. Murano, G. Perelli, and M. Vardi. Reasoning About Strategies: On the Model-Checking Problem. *ACM Trans. Comp. Log.*, 15(4):34:1–42, 2014.

[35] P. Novák and W. Jamroga. Code patterns for agent oriented programming. In *AAMAS*, pages 105–112, 2009.

[36] S. Paul and R. Ramanujam. Imitation in large games. *arXiv preprint arXiv:1006.2992*, 2010.

[37] S. Paul, R. Ramanujam, and S. Simon. Stability under strategy switching. In *Conference on Computability in Europe*, pages 389–398. Springer, 2009.

[38] J. Queille and J. Sifakis. Specification and Verification of Concurrent Programs in Cesar. In *International Symposium on Programming'81*, LNCS 137, pages 337–351, 1981.

[39] M. Rabin and D. Scott. Finite Automata and their Decision Problems. *IBMJRD*, 3:115–125, 1959.

[40] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.

[41] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *AAMAS*, pages 157–164, 2005.

[42] W. van der Hoek and M. Wooldridge. Tractable Multiagent Planning for Epistemic Goals. In *Autonomous Agents and Multiagent Systems'02*, pages 1167–1174, 2002.

[43] S. Vester. Alternating-time temporal logic with finite-memory strategies. In *GandALF*, EPTCS 119, pages 194–207, 2013.

[44] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In *TARK XI*, pages 269–278, 2007.

[45] N. Yadav and S. Sardiña. Reasoning about agent programs using ATL-like logics. In *JELIA*, LNCS 7519, pages 437–449, 2012.