Natural Strategic Ability

Wojciech Jamroga[a,b], Vadim Malvone[c], Aniello Murano[d]

[a]*Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland*
[b]*Interdisciplinary Centre for Security, Reliability and Trust, SnT, University of Luxembourg*
[c]*Université d'Evry, France*
[d]*Università degli Studi di Napoli Federico II, Italy*

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 rather unlikely to come up with such plans in the first place. A similar concern applies to artificial agents with limited memory and/or computational power. In this paper, we adopt the view of bounded rationality, and look at "simple" strategies in specification 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.

After that, we turn to the broader issue of natural strategic abilities in concurrent games with LTL-definable winning conditions, and study a number of decision problems based on surely winning and Nash equilibrium. We show that, by adopting the view of bounded rationality based on natural strategies, we significantly decrease the complexity of *rational verification* for multi-agent systems.

Keywords: multi-agent systems, strategic ability, alternating-time temporal logic, bounded rationality, model checking, concurrent games, rational verification.

1. Introduction

More and more systems nowadays involve social as much as technological aspects, and even those that focus on technology are often based on distributed components exhibiting self-interested, goal-directed behavior. The field of *multi-agent systems (MAS)* studies the whole spectrum of phenomena ranging from agent architectures to communication and coordination in agent groups to agent-oriented software engineering. The theoretical foundations are mainly based on game theory and formal logic.

Reasoning about strategies. As the systems around us become more complex, and at the same time more autonomous, the need for unambiguous specification and automated verification rapidly increases. Many relevant properties of multi-agent systems refer to *strategic abilities* of agents and their groups. For example, functionality requirements can be often understood in terms of the users' ability to complete the

selected tasks. Similarly, many security properties boil down to the inability of the intruder to obtain his goals. *Logics of strategic reasoning* provide powerful tools to reason about such aspects of MAS [12, 119, 113, 79, 45, 90]. The logics allow to express properties of agents' behavior and its dynamics, driven by individual and collective goals of the agents. 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* [47, 103], which makes it possible to verify the correct behavior of a multi-agent system using recently developed automated tools [86, 85, 43, 44].

A fundamental contribution in this field is *alternating-time temporal logic* ATL and its syntactic extension ATL* [12]. Formulas of ATL are usually interpreted over *concurrent game structures (CGS)* which are labeled state-transition systems that model synchronous interaction among agents. For example, the ATL formula $\langle\langle c\rangle\rangle \mathsf{F}\mathsf{ticket}$ may be used to express that the customer c can ensure that he will eventually obtain a ticket, regardless of the actions of the other agents. The specification holds if c has a strategy whose every execution path satisfies $\mathsf{F}\mathsf{ticket}$. Clearly, it captures an important functionality requirement for any ticket vending machine. Similarly, $\langle\langle v, crc\rangle\rangle \mathsf{G}(\mathsf{voted}_{\mathsf{v,i}} \leftrightarrow \mathsf{AF}\mathsf{paid}_{\mathsf{crc,v}})$ says that the voter and the coercer have a collective strategy to ensure that the coercer will pay out the prearranged bribe whenever v has voted for the indicated candidate i. This is obviously an undesirable property for most voting systems, as it allows to establish a vote selling/buying scheme between the coercer and the voter.

Natural strategies. Following the game-theoretic model of multi-step interaction, strategies in MAS 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 [45, 90]) 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. The approach makes sense from the mathematical point of view, and might be appropriate to reason about strategic abilities of a machine (robot, computer program) with extensive computational power. We claim, however, that it is unrealistic 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 is to come up with the strategy on his own.[1] A similar concern can be raised for strategic abilities of artificial agents with limited memory and/or computational power, such as simple robots, sensors in an autonomous sensor network, and components of Internet of Things.

In this paper, we propose that "natural" ability should be based on strategies whose complexity does not exceed a given bound. To reason about such strategies, we introduce NatATL, a logic that updates 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

[1] Paraphrasing the classical usability desiderata of Nielsen [95], it should be *easy to obtain*, *easy to remember*, *efficient to use*, *preventing errors*, and *satisfying*.

measure the complexity of strategies, we assume that they are represented by lists of condition-action rules. For memoryless strategies, conditions are Boolean propositional formulas. For strategies with recall, conditions are given as regular expressions over Boolean propositional formulas. We discuss this choice of representation in more detail in Section 4.5.

Going back to our motivating examples, $\langle\langle c\rangle\rangle \mathsf{F}\,\mathsf{ticket}$ is satisfied in ATL even if the simplest successful strategy for c includes millions of clauses, each based on a complicated Boolean condition. In contrast, $\langle\langle c\rangle\rangle^{\leq 10} \mathsf{F}\,\mathsf{ticket}$ in NatATL expresses that the customer can obtain a ticket by means of a strategy of complexity at most 10, which seems much more appropriate as the functionality requirement. Similarly, $\neg\langle\langle v, crc\rangle\rangle^{\leq 50} \mathsf{G}(\mathsf{voted}_{\mathsf{v,i}} \rightarrow \mathsf{AF}\,\mathsf{paid}_{\mathsf{crc,v}})$ might be enough to prevent vote buying: if the selling strategy is too complicated, most voters will neither be able to follow it, nor trust the coercer in this capacity.

Games of natural strategic ability. Besides redefining strategy-based requirements, *natural strategies* provide a new, different view of concurrent multi-player games. In concurrent games, the overall behavior of the system emerges from interaction of individual behaviors driven by individual goals of agents [116, 128, 63]. In the new view, the emergence results from *simple* individual play. We observe that the standard game-theoretic solution concepts, such as dominant strategies or Nash equilibrium, can be applied to study the resulting patterns of interaction. Moreover, they can be used to reason about the properties of interaction between *rational agents* whose resources are limited, especially with respect to the plans that they can identify, represent, and execute. This is somewhat similar to the well-known work on *bounded rationality*. However, the bounds in our case are of a different nature than normally considered in game theory, as they refer to the sophistication of strategies rather than to the computational hardness of producing a strategy.

Motivation. Our motivation is, first of all, conceptual. We propose a framework that, in our opinion, is better suited to capture strategic abilities of realistic agents than formalisms based on a combinatorial notion of a strategy. The framework can be used to characterize, reason about, and verify abilities of players whose capability of producing and handling plans is limited. It also provides a foundation to define metrics for various functionality, security, and usability properties in multi-agent systems. Moreover, the focus on simple strategies may ultimately make verification of some systems more feasible. This brings us to the second motivation behind our work.

Several recent papers within AI have studied verification of systems, based on their *bounded* characteristics. The starting point was an observation that model checking is somewhat resistant to practical applications, even when the theoretical (asymptotic) complexity seems appealing, like the polynomial time results for model checking CTL, CTLK, and ATL. Still, in realistic scenarios, the space of states, transitions, available choices, strategies etc. is immensely large. When coupled with the machinery required to deal with infinite sets of infinite computation paths in the system, this results in algorithms and tools that do not scale up well enough. As a consequence, a number of studies have been published, concerned with properties of systems in finite computation paths, and hence using finite trees as the behavioral model of the system [52, 53, 29]. Similarly, one can look at the properties to be model-checked, and focus only on those

that can be demonstrated in a finite time horizon [82, 92, 13]. In this paper, we propose that the scope of verification can be restricted also at the level of strategic choices taken by agents in the system. For the moment, we offer only theoretical complexity results. Ultimately, we want to use the resulting variants of model checking and game solving to analyze realistic interaction scenarios.

Technical contribution. The main contribution of this paper consists of the following points:

1. We propose the concept of *natural strategies*, based on an intuitive representation of conditional plans. We also propose how to measure the sophistication (or complexity) of such strategies;

2. We define NatATL, a variant of alternating-time temporal logic to reason about natural strategic ability, and we study the complexity of NatATL model checking. We consider two main cases here: the abilities of agents playing memoryless strategies, and agents using strategies with recall of the past;

3. For natural strategies with recall, we show that their relationship to memoryless strategies is more intricate than normally in ATL. Moreover, we investigate the relationship of natural strategies with recall to the representation of strategies based on input/output automata (i.e., finite transducers);

4. Finally, we investigate several decision problems for natural abilities of agents in concurrent games with LTL-definable winning conditions. We specifically look at problems related to *surely winning* and *Nash equilibrium*. We define the problems formally and establish their computational complexity.

Structure of the article. We begin by discussing the related work (Section 2) and recalling the main concepts behind logical reasoning about strategic ability (Section 3). Then, in Section 4, we introduce the concept of natural memoryless strategies, show how to measure their complexity, and define NatATL as a variant of alternating-time temporal logic for reasoning about existence of such strategies. We study the complexity of model checking for the new logic in Section 5. In Sections 6 and 7, we turn to natural strategic abilities of agents with recall of the past. We define natural strategies with memory together with the corresponding variant of NatATL, study their relationship to memoryless strategies and to finite-memory strategies represented by deterministic finite transducers, and establish some complexity results for the model checking problem. The second part of the paper is concerned with natural abilities in concurrent games with players' objectives defined by formulas of linear time logic (LTL). We focus on two solution concepts: surely winning and Nash equilibrium. Some relevant decision problems are formally defined and discussed in Section 8; we study their complexity in Section 9. Finally, we conclude in Section 10.

Previous version of the article. Some of the ideas and results discussed here have been already presented in a preliminary form in the conference paper [78]. Here, we extend the conference version with a study of natural ability for rational players in concurrent games (Sections 8 and 9, roughly the second half of this article). We also pay special attention to motivating examples, and make a proper case for the applicability

of natural strategies in specification of multi-agent systems. Finally, we have added some new results, as well as expanded and revised a number of proofs in the part on logical reasoning. Most importantly, we revise the erroneous claim from [78] concerning the relationship between natural strategies with recall and finite transducers, and characterize the relationship correctly. We also correct some results concerning the complexity of decision problems for natural strategies with recall.

2. Related Work

Logical reasoning about strategies. Strategic reasoning has been the subject of intensive research within multi-agent systems and AI. Variants of ATL and ATL* are probably the most popular logic-based approaches to formalizing strategic abilities of agents and coalitions. The research on alternating-time temporal logic can be roughly divided into the computational and the conceptual strand. The former is focused on the way in which ATL and its extensions can be used for verification of multi-agent systems – in particular, what is the complexity of model checking, and how one can overcome the inherent difficulties. An interested reader is referred to [35] for an overview, and to [12, 113, 118, 76, 84, 39, 54, 17, 77, 16, 87] for more specific results. Analogous research have been conducted for the more expressive language of Strategy Logic [45, 93, 90, 71]. Satisfiability for the perfect information variants of ATL and SL has been investigated in [59, 126, 109, 89, 91].

The conceptual strand originally emerged in quest of the "right" semantics for ability under imperfect information, see [3, 40] for an introduction. ATL was combined with epistemic logic [120, 121, 1, 74], and several semantic variants were defined that match various possible interpretations of ability [113, 79, 74]. Imperfect information strategies have been also studied within the SL framework [18, 21]. Moreover, many conceptual extensions have been considered, e.g., with explicit reasoning about strategies [117, 127], rationality assumptions and solution concepts [122, 41], coalition formation and negotiation [34], opponent modeling and action in stochastic environments [73, 38, 112, 111], fairness conditions and prompt temporal operators [42, 13], reasoning about irrevocable plans and interplay between strategies of different agents [2, 33], and agents with bounded resources (for the latter, see the references mentioned below).

Strategic abilities of resource-bounded agents. The main motivation behind this article comes from the second, conceptual strand. Some works that are close in spirit to our proposal concern modeling, specification, and reasoning about strategies of agents with bounded mental capabilities. Most importantly, [8, 10, 36, 37, 5, 9] extend temporal and strategic logics to handle agents with bounded resources, such as time, memory, money, etc. More specifically, [4] studies strategic properties of agents with limited memory. Issues related to bounded memory are also investigated in [15, 70, 123, 61]. Still, all those works consider the standard combinatorial notion of a strategy, and are mostly concerned with the decidability frontier for reasoning about different classes of resource dynamics.

Representation of strategies. Another relevant group of papers concerns various representations of strategies. This category includes extensions of ATL with explicit rea-

soning about actions and strategies [117, 1, 127, 66], as well as logics that combine features of temporal and dynamic logic [64, 97]. A variant of STIT logic that enables reasoning about strategies and their performance in the object language was also investigated in [55]. A representation of finite-memory strategies by input/output automata was used in [124]. Moreover, some recent works consider minimization of strategies represented by decision diagrams, in order to obtain more human-friendly descriptions [30, 31]. Again, the above papers take the standard, combinatorial representation of strategies as their starting point. In contrast, we start with an intuitive structure for representing conditional plans, well known in AI, and study how it changes the specification and verification of agents' abilities.

In fact, our representations of strategies are very close to conditional plans used in classical approaches to automated planning [106, 58] and agent-oriented programming [67, 68, 26]. Reasoning about agent programs with formulas of strategic logics was also investigated in [25, 6, 7, 48, 129]. We will comment in more detail on the inspiration behind our representations in Section 4.5. At this point we only mention that, unlike the literature on planning and agent-oriented programs, we do not focus on practical application-driven languages for specification of plans and their interaction with other mental attitudes. Instead, we study an abstract representation class that seems to "distill" the most relevant features of the practical solutions.

Bounded rationality. Closer to game theory, our work is related to the general concept of *bounded rationality* and to the idea of *rational verification*. The bounded rationality view [105, 114] assumes that the players have bounded computational power when reasoning about their choices. In particular, some researchers investigated the impact of players' bounded memory in games [81, 70, 22]. However, unlike in our proposal, the bounds are considered from a purely mathematical point of view, and not in relation to the practical ease of the underlying mental operations. Finally, rational verification (also known as *equilibrium checking*) is an extension of model checking to verify behavior of agents limited to rational choices. Typically, this kind of verification is based on Nash equilibrium, and applied to concurrent games with players' objectives defined by formulas of linear time logic [116, 128, 63]. Preliminary approaches to similar analysis based on other solution concepts were presented in [65, 41, 100, 99]. The last part of our paper comes close; however, we study the outcome of natural bounded strategies, and obtain different complexity results.

Bounded approaches to verification. Bounded characteristics of interaction has been studied for conceptual and computational reasons. We already discussed the conceptual approaches, focusing on reasoning about *bounded agents*. The computational strand concentrates on *bounded reasoning* about agents, with the hope to obtain algorithms and tools that scale up well in practical applications. One can trace back the idea to the concept of bounded model checking [23], later applied also to properties of multi-agent systems [101, 80]. More recently, a number of papers investigated model checking and synthesis for LTL and linear dynamic logic in finite computation paths [52, 53, 29, 19]. Similar research considered verification of properties that can be demonstrated in a finite time horizon [82, 92, 13]. This paper is not directly related to bounded verification, but can potentially contribute to advancing the field, by considering yet another angle of bounded search – namely, bounded search through the space of strategies.

Summary. There has been much research in the last 20 years on the right semantics of strategic ability. Moreover, bounded rationality is an important topic within game theory. The two strands come together in so called "rational verification," where model checking techniques are adapted to models of rational behavior. Bounded approaches to verification of multi-agent systems also bear some relevance for our research.

We have summarized the most important ideas above. However, none of the 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.

3. Preliminaries: What Agents Can Achieve

We begin by recalling the syntax and semantics of alternating-time temporal logic (ATL) which has been used with great success by the MAS community to formalize reasoning about strategic abilities of agents and coalitions.

3.1. Expressing Abilities of Agents and Coalitions

Alternating-time temporal logic (ATL, for short) [11, 12] generalizes the 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 act, formula γ is satisfied. The formulas make use of temporal operators: "X" ("next"), U ("strong until"), and W ("weak until"). Formally, let $\mathbb{A}\mathrm{gt}$ be a finite set of agents, and $Prop$ a countable set of atomic propositions. The language of ATL is defined as follows:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \mathsf{X}\, \varphi \mid \langle\langle A\rangle\rangle \varphi\, \mathsf{U}\, \varphi \mid \langle\langle A\rangle\rangle \varphi\, \mathsf{W}\, \varphi.$$

where $A \subseteq \mathbb{A}\mathrm{gt}$ and $\mathsf{p} \in Prop$. Derived Boolean connectives and constants $(\vee, \rightarrow, \top, \bot)$ are defined as usual. "Now or sometime in the future" can be defined as $\mathsf{F}\varphi \equiv \top\, \mathsf{U}\, \varphi$, and "always from now on" as $\mathsf{G}\varphi \equiv \varphi\, \mathsf{W}\, \bot$. The path quantifier "for all paths" can be defined as $\mathsf{A}\gamma \equiv \langle\langle\emptyset\rangle\rangle\gamma$. Finally "there is a path" can be defined as dual to A, i.e., $\mathsf{E}\,\varphi\,\mathsf{U}\,\psi \equiv \neg\mathsf{A}(\neg\psi)\,\mathsf{W}\,(\neg\varphi \wedge \neg\psi)$ and $\mathsf{E}\,\varphi\,\mathsf{W}\,\psi \equiv \neg\mathsf{A}(\neg\psi)\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)$.

Example 1 (Ticket machine). *The* ATL *formula* $\langle\langle c\rangle\rangle\mathsf{F}\mathsf{ticket}$ *expresses that the customer* c *is able to eventually obtain the ticket, no matter what anybody else does and what circumstances arise.*

3.2. Models of Multi-Agent Interaction

The semantics of ATL is defined over concurrent game structures, a variant of synchronous multi-agent transition systems.

Definition 1 (CGS). *A* concurrent game structure (CGS) *is a tuple* $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V\rangle$ *which includes nonempty finite sets of: agents (or players)* $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_{|\mathbb{A}\mathrm{gt}|}\}$*, states* St*, actions* Act*, atomic propositions* $Prop$*, and a propositional valuation* $V : St \rightarrow 2^{Prop}$*. The function* $d : \mathbb{A}\mathrm{gt} \times St \rightarrow 2^{Act}$ *defines the availability of actions in states. The (deterministic) transition function* t *assigns a successor state* $q' = t(q, \alpha_1, \ldots, \alpha_{|\mathbb{A}\mathrm{gt}|})$ *to each state* $q \in St$ *and any tuple of actions*

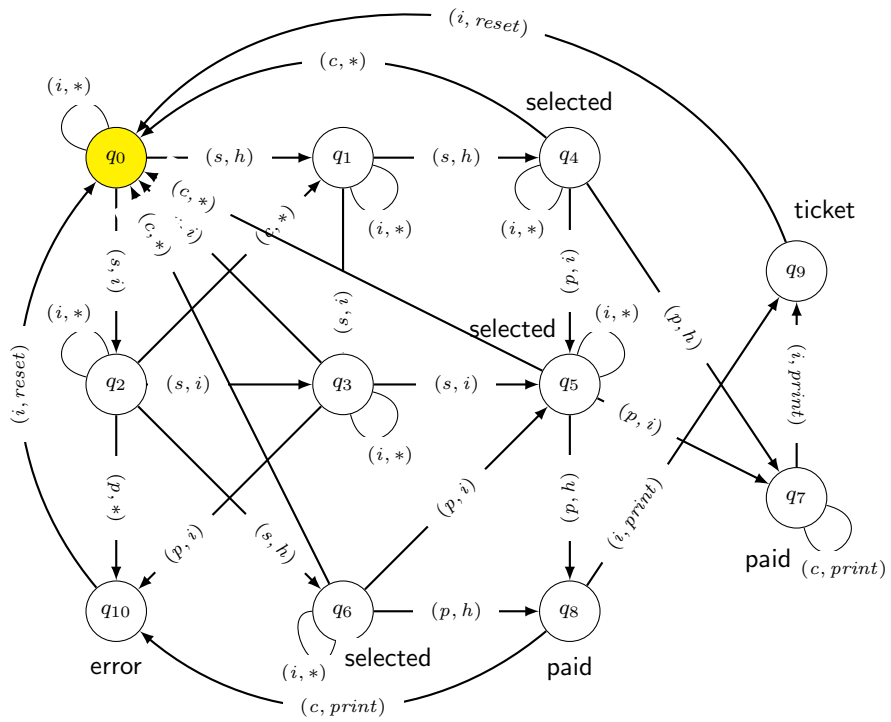

Figure 1: M_{ticket}: a model of the ticket machine

$\alpha_i \in d(a_i, q)$ *that can be executed by* $\mathbb{A}\mathrm{gt}$ *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 choices of group* A *in state* q *by* $d_A(q) = \prod_{a_i \in A} d_{a_i}(q)$*.*

A pointed CGS *is a pair* (M, q_0) *where* M *is a concurrent game structure and* q_0 *a state in* M*.*

Example 2 (Ticket machine). *An example of a CGS for a ticket vending machine is presented in Figure 1. The model specifies interaction between the customer* (c) *and the program of the machine* (m)*. Initially, the customer can make his selection on the touchscreen (action* $select$*, or* s *for short) or stay idle and do nothing (action* $idle$ *or* i*). The machine, on the other hand, can be run in two different modes: let us call them "helpful" (action* $helpful$*, or* h *for short) and "default" (action* $idle$ *or* i*). When the customer has finished the selection (which is indicated by the atomic proposition* selected*), he can proceed to payment (action* pay*, or* p *for short) or cancel the operation (action* $cancel$ *or* c*). After the payment, the system duly prints the ticket (action* $print$*), and resets the process to its initial state* q_0 *(action* $reset$*).*

Due to a flawed implementation, it is possible to execute actions p *or* c *also in states where it should not be, leading to haphazard results. For example, cancelling*

after payment leads to the error state q_{10}, paying in q_2 or q_3 (before the selection is made) leads to the error state, too, and cancelling in q_2 leads to q_1.

3.3. Strategies and Their Outcomes

A *path* $\lambda = q_0 q_1 q_2 \dots$ 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. By Λ_M, we denote the set of all the paths in M, and by $\Lambda_M(q)$ all the paths in M starting in q. Similarly, a *history* $h = q_0 q_1 q_2 \dots q_n$ 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 $H_M = St^+$ the set of all the histories in model M. We will omit the subscripts whenever they are clear from the context.

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. Formally, it can be represented by a function $s_a : St \to Act$ satisfying $s_a(q) \in d_a(q)$ for each $q \in St$.[2] Note that this is the memoryless (a.k.a. positional) notion of a strategy. We will follow the notation proposed by Schobbens [113], and refer to such strategies as *r*-strategies. A *collective strategy* s_A for coalition $A \subseteq \mathbb{A}\mathrm{gt}$ is a tuple of individual strategies, one per agent from A. The set of all such strategies is denoted by Σ^r_A. A collective strategy of all the agents in $\mathbb{A}\mathrm{gt}$ is sometimes called a *strategy profile*.

Let $s_A[a]$ denote the strategy of agent $a \in A$ selected from s_A. Moreover, function $out(q, s_A)$ returns the set of all the paths that can result from the execution of strategy s_A from state q. Formally:

$out(q, s_A) = \{\lambda = q_0, q_1, q_2 \dots \mid q_0 = q$ and for each $i = 0, 1, \dots$ there exists $(\alpha^i_{a_1}, \dots, \alpha^i_{a_k})$ such that $\alpha^i_a \in d_a(q_i)$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha^i_a = s_A[a](q_i)$ for every $a \in A$, and $q_{i+1} = o(q_i, \alpha^i_{a_1}, \dots, \alpha^i_{a_k})\}$.

3.4. Semantics of ATL

Given a CGS M and a state $q \in St$, the semantics of ATL is defined by the following clauses:

$M, q \models_r \mathsf{p}$ iff $\mathsf{p} \in V(q)$, for $\mathsf{p} \in Prop$;

$M, q \models_r \neg\varphi$ iff $M, q \not\models_r \varphi$;

$M, q \models_r \varphi_1 \wedge \varphi_2$ iff $M, q \models_r \varphi_1$ and $M, q \models_r \varphi_2$;

$M, q \models_r \langle\langle A\rangle\rangle \mathsf{X}\,\varphi$ iff there is a strategy $s_A \in \Sigma^r_A$ such that, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[1] \models_r \varphi$;

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

[2] Observe that function-based representations of strategies are motivated by mathematical elegance and generality, rather than similarity to the way humans actually deliberate and act. We will sometimes call such strategies *combinatorial* because of that.

$M, q \models_r \langle\langle A \rangle\rangle \varphi \,\mathsf{W}\, \psi$ iff there is a strategy $s_A \in \Sigma_A^r$ such that, for each path $\lambda \in out(q, s_A)$, we have either that $M, \lambda[i] \models_r \psi$ for some $i \geq 0$ and $M, \lambda[j] \models_r \varphi$ for all $0 \leq j < i$, or that $M, \lambda[i] \models_r \varphi$ for all $i \geq 0$.

Example 3 (Ticket machine). *Consider the ticket machine in Example 2. We will now demonstrate that* $M_{ticket}, q_0 \models \langle\langle c \rangle\rangle \mathsf{F}\,\mathsf{ticket}$. *That is, in* q_0*, the customer can ensure that he will eventually obtain a ticket. An example strategy that demonstrates the ability is:* $s_c(q_0) = s, s_c(q_1) = s, s_c(q_2) = c, s_c(q_3) = s, s_c(q_4) = p, s_c(q_5) = p, s_c(q_6) = c, s_c(q_7) = i, s_c(q_8) = i, s_c(q_9) = i, s_c(q_{10}) = i$.

Memoryless vs. memoryful semantics of ATL. ATL was originally introduced with a semantics based on *perfect recall strategies* of type $s_a : St^+ \to Act$, such that $s_a(h) \in d_a(last(h))$ for every $h \in H$. We will sometimes refer to such strategies as *R*-strategies; the set of such strategies for coalition A is denoted by Σ_A^R.

The memoryful semantics of ATL, defined by relation $\models_R$, can be obtained from $\models_r$ by replacing all occurrences of Σ_A^r with Σ_A^R. The following result is well-known, and based on the fact that memoryless strategies suffice for safety and reachability goals in games of perfect information.

Proposition 1 ([12, 113]). *For a model* M*, state* q *in* M*, and* ATL *formula* φ*, we have:*

$$M, q \models_r \varphi \quad \textit{iff} \quad M, q \models_R \varphi.$$

4. Reasoning about Natural Ability

In this section we introduce NatATL, a logic for reasoning about natural strategic ability. We argue that the notion of ability, captured by ATL, misses an important aspect of how humans reason about the world. As a remedy, we propose a modification of the logic that allows to take into account the combinatorial limitations of human reasoning.

An important motivation for this work comes from *usability concerns* in functionality requirements. 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.

4.1. Natural Strategies and Their Complexity

We start by defining the notion of *natural memoryless strategy* (or *nr*-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. Formally, let $\mathcal{B}(\Xi)$ be the set of Boolean formulas over alphabet Ξ. We represent natural strategies by *lists of guarded actions*, i.e., sequences of pairs $(\varphi_i,\ \alpha_i)$ such that: (1) $\varphi_i \in \mathcal{B}(Prop)$, and (2) $\alpha_i \in d_a(q)$ for every $q \in St$ such that $q \models \varphi_i$. That is, φ_i is a propositional condition on states of the CGS,

and α_i is an action available to agent a in every state where φ_i holds.[3] Moreover, we assume that the last pair on the list is $(\top,\ \alpha)$ for some $\alpha \in Act$, i.e., the last rule is guarded by a condition that will always be satisfied.

The set of all natural memoryless strategies of agent a is denoted by Σ_a^{nr}. By $length(s_a)$, we denote the number of guarded actions in s_a. Moreover, $cond_i(s_a)$ denotes the ith guard (condition) on the list, and $act_i(s_a)$ the corresponding action. Finally, $match(q, s_a)$ is the smallest $i \leq length(s_a)$ such that $q \models cond_i(s_a)$ and $act_i(s_a) \in d_a(q)$. 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. Additionally, $dom(\varphi) = \{\mathsf{p} \in Prop \mid \mathsf{p} \in \varphi\}$ stands for the set of atomic propositions that appear in condition φ, and $dom(s_a) = \bigcup_{i=1,\ldots,length(s_a)} dom(cond_i(s_a))$ denotes the propositions occurring in s_a.

A *collective natural strategy* for agents $A = \{a_1, \ldots, a_{|A|}\}$ is a tuple of individual natural strategies $s_A = (s_{a_1}, \ldots, s_{a_{|A|}})$. The set of such strategies is denoted by Σ_A^{nr}. 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:

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

We emphasize that the outcome of s_A collects *all* the paths consistent with s_A. In particular, the opponents are not assumed to play a natural strategy; in fact, they are not assumed to play any strategy at all.

Example 4 (Ticket machine). *The strategy in Example 3 guarantees that the customer will obtain a ticket, but it is unrealistic to expect human customers that they follow such a specification, let alone come up with it on their own. Moreover, it unnecessarily prescribes action $cancel$ in state q_2, a fact which is buried in the table-like description of the strategy function, and therefore very difficult to spot. On the other hand, consider the following specification in the form of an nr-strategy:*

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

It says that the customer selects a ticket whenever the selection has not been made (nor actually already got the ticket, or at least made the payment etc.). After making the selection, he does the payment, and else he stays idle. The strategy, and the transitions consistent with it, are highlighted in Figure 2. The outcome of the strategy from state q_0 consists of all the paths that start in q_0 and can be composed of the highlighted transitions.

[3] In fact, it suffices to require that $\alpha_i \in d_a(q)$ for every $q \in St$ such that $q \models \neg\varphi_1 \wedge \cdots \wedge \neg\varphi_{i-1} \wedge \varphi_i$, see the operational semantics of natural strategies in the next two paragraphs.

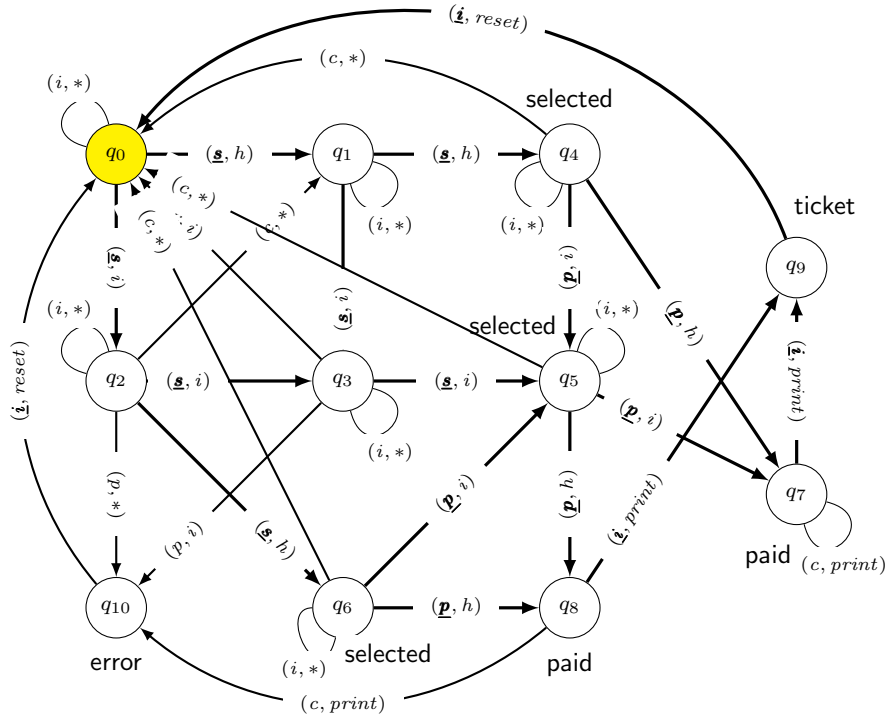

Figure 2: A strategy for buying a ticket

By $compl(s_a)$, we denote the complexity, or equivalently the size, 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 $(\mathcal{B}(Prop) \times Act)^+$, e.g.:

Number of used propositions: $compl_{\#}(s_a) = |dom(s_a)|$;
Largest condition: $compl_{\max}(s_a) = \max\{|\varphi| \mid (\varphi, \alpha) \in s_a\}$;
Total size of the representation: $compl_{\Sigma}(s_a) = \sum_{(\varphi,\alpha)\in s_a} |\varphi|$

with $|\varphi|$ being the number of symbols in φ, without parentheses. 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. That is, unless explicitly specified, we will assume $compl(s_a) = compl_{\Sigma}(s_a)$.

Example 5 (Ticket machine). *Consider the nr-strategy s_c from Example 4. If we look at the number of used propositions, we have that* $compl_{\#}(s_c) = |\{\mathsf{ticket}, \mathsf{selected}, \mathsf{paid}\}, \mathsf{error}| = 4$. *If we consider the largest condition instead, we have* $compl_{\max}(s_c) = 11$. *Finally, if we use the total size of the representation, we get* $compl_{\Sigma}(s_c) = 13$.

Interestingly, an equivalent natural strategy with $compl_\Sigma(s_c) = 5$ exists:
((paid, $idle$); (ticket, $idle$); (error, $idle$); (selected, pay); ($\top$, $select$)).

The complexity of a natural collective strategy $s_A = (s_{a_1}, \ldots, s_n)$ can be defined analogously:

- $compl_{\#}(s_A) = |\bigcup_{i=1,\ldots,n} dom(s_{a_i})|$,
- $compl_{\max}(s_A) = \max_{i=1,\ldots,n}\{compl_{\max}(s_{a_i})\}$, and
- $compl_\Sigma(s_A) = \sum_{i=1,\ldots,n} compl_\Sigma(s_{a_i})$.

Unless explicitly specified, we assume $compl(s_A) = compl_\Sigma(s_A)$.

4.2. A Logic for Natural Strategies

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 in 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"). Thus, the language of NatATL can be definen by the following grammar:

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

where $A \subseteq \mathbb{A}\mathrm{gt}$, $k \in \mathbb{N}$, and $\mathsf{p} \in Prop$.

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.

Example 6 (Ticket machine). *The* ATL *specification* $\langle\langle c\rangle\rangle\mathsf{F}\mathsf{ticket}$ *says that the customer has a conditional plan such that, no matter what the other agents do, the ticket will eventually be issued. Does that mean that the customer* is able *to obtain the ticket? It has been argued in [79] that the strategic ability to enforce a property should imply that the agent can* identify *and* execute *the right strategy. If the strategy is too complex, people will not be capable of using it. Thus, instead of* $\langle\langle c\rangle\rangle\mathsf{F}\mathsf{ticket}$*, one should require that the ticket machine satisfies* $\langle\langle c\rangle\rangle^{\leq k}\mathsf{F}\mathsf{ticket}$ *for a reasonably low* k*.*

Example 7 (Gaming). *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 the game for player* a *is* k *iff the game satisfies the formula* $\langle\langle a\rangle\rangle^{\leq k}\mathsf{F}\mathsf{win} \wedge \neg\langle\langle a\rangle\rangle^{\leq k-1}\mathsf{F}\mathsf{win}$.

Example 8 (Vote buying). *Coercion-resistance is an important property of voting systems [20]. A system is coercion-resistant if the coercer cannot influence the way the voter votes, even if the voter is willing to cooperate with the coercer. The latter assumption is relevant especially in cases of* vote buying*. The* ATL *formula* $\langle\langle v, crc\rangle\rangle\mathsf{G}(\mathsf{voted}_{\mathsf{v,i}} \rightarrow \mathsf{AF}\,\mathsf{paid}_{\mathsf{crc,v}})$ *says that the voter and the coercer have a collective strategy to ensure that, whenever* v *has voted for the indicated candidate* i*, the coercer will*

pay out the prearranged bribe. However, the formula does not take into account how complex is the behavior that the agents must use in order to successfully sell/buy the vote.

Similarly to Example 7, one can define the level of coercion resistance as the smallest k such that $\langle\langle v, crc\rangle\rangle^{\leq k}\mathsf{G}(\mathsf{voted}_{\mathsf{v,i}} \rightarrow \mathsf{AF}\,\mathsf{paid}_{\mathsf{crc,v}}) \wedge \neg\langle\langle v, crc\rangle\rangle^{\leq k-1}\mathsf{G}(\mathsf{voted}_{\mathsf{v,i}} \rightarrow \mathsf{AF}\,\mathsf{paid}_{\mathsf{crc,v}})$.

4.3. 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:

- $M, q \models_{nr} \mathsf{p}$ iff $\mathsf{p} \in V(q)$, for $\mathsf{p} \in Prop$;
- $M, q \models_{nr} \neg\varphi$ iff $M, q \not\models_{nr} \varphi$;
- $M, q \models_{nr} \varphi_1 \wedge \varphi_2$ iff $M, q \models_{nr} \varphi_1$ and $M, q \models_{nr} \varphi_2$;
- $M, q \models_{nr} \langle\langle A\rangle\rangle^{\leq k}\mathsf{X}\,\varphi$ iff there is a strategy $s_A \in \Sigma^{nr}_A$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[1] \models_{nr} \varphi$;
- $M, q \models_{nr} \langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{U}\,\psi$ iff there is a strategy $s_A \in \Sigma^{nr}_A$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[i] \models_{nr} \psi$ for some $i \geq 0$ and $M, \lambda[j] \models_{nr} \varphi$ for all $0 \leq j < i$.
- $M, q \models_{nr} \langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{W}\,\psi$ iff there is a strategy $s_A \in \Sigma^{nr}_A$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have either that $M, \lambda[i] \models_{nr} \psi$ for some $i \geq 0$ and $M, \lambda[j] \models_{nr} \varphi$ for all $0 \leq j < i$, or that $M, \lambda[i] \models_{nr} \varphi$ for all $i \geq 0$.

Again, we point out that, when evaluating formula $\langle\langle A\rangle\rangle^{\leq k}\gamma$ we do not assumed the opponents to play a natural strategy (bounded or otherwise). This corresponds to the pessimistic approach to evaluation of ability based on "surely winning:" the agents in A win only if they have a strategy that wins against every – even accidental – behavior of the rest of the system.

Example 9 (Ticket machine). *The strategy in Example 4 can be used to demonstrate that* $M_{ticket}, q_0 \models_{nr} \langle\langle c\rangle\rangle^{\leq 9}\mathsf{F}\,\mathsf{ticket}$. *In fact, as we already remarked, the minimal natural strategy to obtain the ticket is of size* 5. *Thus, we also have that* $M_{ticket}, q_0 \models_{nr} \langle\langle c\rangle\rangle^{\leq 5}\mathsf{F}\,\mathsf{ticket} \wedge \neg\langle\langle c\rangle\rangle^{\leq 4}\mathsf{F}\,\mathsf{ticket}$.

We will refer to the logical system (NatATL, $\models_{nr}$) as NatATL$_r$, and analogously for 1NatATL$_r$.

4.4. Some Properties

Before moving on to study the computational complexity of verification for natural strategies, we list several interesting properties of the logic. First, similarly to ATL, NatATL$_r$ is an extension of the branching-time logic CTL. This is because the path quantifier "for all paths" can be defined in NatATL as $\mathsf{A}\gamma \equiv \langle\langle\emptyset\rangle\rangle^{\leq 0}\gamma$, see the following proposition.

Proposition 2. *For every CGS M, state q in M, and* NatATL *formulas φ, ψ, we have that:*

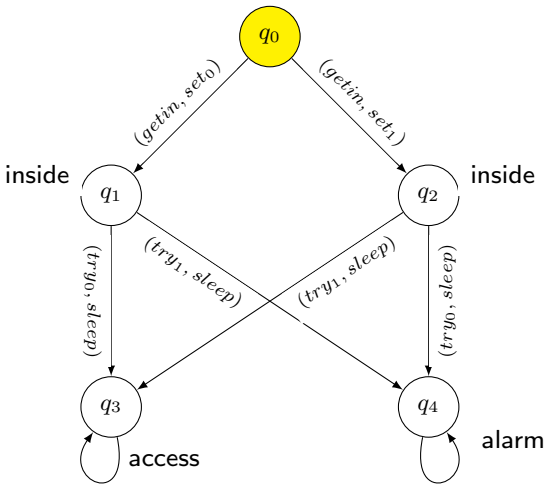

Figure 3: Schobbens' robber: a counterexample to composability of strategies

1. $M, q \models_{nr} \langle\langle\emptyset\rangle\rangle^{\leq 0}\mathsf{X}\,\varphi$ *iff, for every path* $\lambda \in \Lambda_M(q)$*, we have* $M, \lambda[1] \models_{nr} \varphi$*;*
2. $M, q \models_{nr} \langle\langle\emptyset\rangle\rangle^{\leq 0}\varphi\,\mathsf{U}\,\psi$ *iff, for every path* $\lambda \in \Lambda_M(q)$*, we have* $M, \lambda[i] \models_{nr} \psi$ *for some* $i \geq 0$ *and* $M, \lambda[j] \models_{nr} \varphi$ *for all* $0 \leq j < i$*;*
3. $M, q \models_{nr} \langle\langle\emptyset\rangle\rangle^{\leq 0}\varphi\,\mathsf{W}\,\psi$ *iff, for every path* $\lambda \in \Lambda_M(q)$*, we have either that* $M, \lambda[i] \models_{nr} \psi$ *for some* $i \geq 0$ *and* $M, \lambda[j] \models_{nr} \varphi$ *for all* $0 \leq j < i$*, or that* $M, \lambda[i] \models_{nr} \varphi$ *for all* $i \geq 0$*.*

Proof. To see this, it suffices to observe that the empty coalition $\emptyset$ has only one possible strategy – the empty strategy $s_\emptyset = (\,)$. Moreover, $compl(s_\emptyset) = 0$, and $out(q, s_\emptyset) = \Lambda_M(q)$. □

Furthermore, only one direction of the ATL fixpoint equivalences has its counterparts in NatATL; the other one is not valid even for arbitrarily large bounds k.

Proposition 3. *The following formulas of* NatATL *are valid, i.e., they hold for every* M*,* q M*, and* φ*:*

1. $\langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{U}\,\psi \quad\rightarrow\quad \psi \vee \varphi \wedge \langle\langle A\rangle\rangle^{\leq k}\mathsf{X}\,\langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{U}\,\psi$*;*
2. $\langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{W}\,\psi \quad\rightarrow\quad \psi \vee \varphi \wedge \langle\langle A\rangle\rangle^{\leq k}\mathsf{X}\,\langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{W}\,\psi$*.*

Proof. Straightforward: the strategy that validates the left hand side of the implication can be used for both strategic operators on the right hand side. □

Proposition 4. *The following formulas of* NatATL *are* not *valid, even if* k_3 *is given as an arbitrary function of* k_1, k_2*:*

1. $\psi \ \vee \ \varphi \wedge \langle\langle A\rangle\rangle^{\leq k_1}\mathsf{X}\,\langle\langle A\rangle\rangle^{\leq k_2}\varphi\,\mathsf{U}\,\psi \quad \rightarrow \quad \langle\langle A\rangle\rangle^{\leq k_3}\varphi\,\mathsf{U}\,\psi$;

2. $\psi \ \vee \ \varphi \wedge \langle\langle A\rangle\rangle^{\leq k_1}\mathsf{X}\,\langle\langle A\rangle\rangle^{\leq k_2}\varphi\,\mathsf{W}\,\psi \quad \rightarrow \quad \langle\langle A\rangle\rangle^{\leq k_3}\varphi\,\mathsf{W}\,\psi$.

Proof. We show a counterexample to (1) in the special case of F. The other case is analogous.

Consider model M_{rob} in Figure 3, which is a variation of the "robber vs. bank" CGS from [113]. We have that $M_{rob}, q_1 \models_{nr} \langle\langle 1\rangle\rangle^{\leq 1}\mathsf{F}\mathsf{access}$, the right strategy being $s_2 = ((\top, try_0))$. Similarly, $M_{rob}, q_2 \models_{nr} \langle\langle 1\rangle\rangle^{\leq 1}\mathsf{F}\mathsf{access}$ due to strategy $s'_2 = ((\top, try_1))$. But that also means that $M_{rob}, q_0 \models_{nr} \langle\langle 1\rangle\rangle^{\leq 1}\mathsf{X}\,\langle\langle 1\rangle\rangle^{\leq 1}\mathsf{F}\mathsf{access}$. On the other hand, $M_{rob}, q_0 \not\models_{nr} \langle\langle 1\rangle\rangle^{\leq k}\mathsf{F}\mathsf{access}$ for all $k \in \mathbb{N}$. □

The counterexample illustrates an important idea behind natural strategies. It is not intuitive for humans to phrase our plans in terms of states of the system, typically generated by valuations of dozens (or even hundreds) of variables and attributes. Instead, we identify situations by their higher-level properties. In a way, we see the world through the vocabulary that we can use to describe it. On the technical level, propositional formulas provide an abstraction of the state space. Strategies assign actions to the abstract states and their subsets.

4.5. Discussion and Remarks

Which representation is "most natural" for strategies of humans and simple artificial agents is, to an extent, a matter of opinion and personal preference. Below, we offer some arguments to justify our choice of representations based on condition-action rules. The choice can be motivated by four important factors:

1. Conditions based on simple properties of states provide a natural abstraction of the state space. Typically, the state space of a MAS is huge, and arises from a combination of state/transition spaces of multiple distributed processes. Such an approach provides a compositional methodology for modeling, but also results in structures that suffer from combinatorial explosion. On the other hand, states in a model are usually labeled by a small number of atomic propositions, selected by the modeler to map the most relevant properties, and often designed to match the perception of the agents. Thus, the set of propositions that appear in the model provides a sparse vocabulary of relevant concepts, that are likely to be used by the agents when phrasing their plans.

   We note that this is consistent with the classical approaches to commonsense reasoning [49] and automated planning [58].

2. The overall structure of a natural strategy is built on a simple rule-based representation. Condition-action rules are very well known in knowledge representation and AI. Their applications include expert systems [72], classical approaches to AI programming [106], such as Horn clauses in Prolog and condition/action lists in Lisp, representation of plans in automated planning (sequential plans, hierarchical plans, and especially conditional plans) [58], and agent plans in agent-oriented programming [67, 27]. Closely related ideas have been extensively used

in robotics, cf. e.g. teleo-reactive programs [96] and the well-known subsumption architecture [102].

Other, more abstract representations of complex decisions, such as decision trees and decision lists [106], are also built on a similar intuition.

3. In order to be operational, plans should be of manageable complexity. This is in line with game-theoretic approaches to bounded rationality [105, 114], as well as some works on usability [95] and the psychology of planning [94].

4. Last but not least, our proposal is consistent with the empirircal research on *human concept learning* [28]. In particular, the results in [56] showed that the subjective difficulty of a concept according to human subjects is often directly proportional to its Boolean complexity, i.e., the length of the shortest logically equivalent propositional formula. The resulting model was also successfully used in a recent work on analysis of social norms to explain the impact of norm complexity on the resulting behavior [108, 107]. This strongly suggests that humans characterize situations by Boolean combinations of simple, primitive concepts. In consequence, it seems that our choice of representations for strategies, and the way we measure their complexity, indeed closely corresponds to how humans perceive the world and reason about it.

An interesting question is how to practically construct natural strategies. In a natural strategy, the first action whose condition holds is chosen. One possibility is to use a kind of "defeasible" specification of behavior, starting with the most specific conditions, followed by the "default," most general option. The opposite approach – most general conditions first – also makes sense, as it minimizes the number of steps needed to find the matching precondition, and hence optimizes the execution of the strategy. Finally, we observe that natural strategies are in fact *decision lists*, which are a special class of *decision trees*. Extending our formalism to full decision trees (or perhaps even decision DAGs as in Binary Decision Diagrams), and applying information-theoretic algorithms that construct an optimal tree for a given behavior, seems an interesting direction for future research, parallel to the work in [31].

5. Model Checking for NatATL

In this section we study algorithms and the complexity of the model checking problem for NatATL with *nr*-strategies, i.e., NatATL$_r$. We consider two cases: one in which the bound k on the size of natural strategies is assumed to be constant, and the more general case where k is variable and a parameter of the problem. For the former case, we prove that the problem is polynomial in the size of the model. For the latter, model checking becomes $\mathbf{\Delta_2^P}$-complete.[4] Moreover, it is $\mathbf{NP}$-complete for simple formulas

[4] $\mathbf{\Delta_2^P} = \mathbf{P^{NP}}$ is the class of problems solvable in polynomial time by a deterministic Turing machine making adaptive calls to an oracle for problems in $\mathbf{NP}$.

with only one strategic operator, i.e., formulas of 1NatATL$_r$. A concise table summarizing all the complexity results for model checking NatATL is presented at the end of the paper (Section 10, Figure 16).

The results and the proofs proposed in this section have been inspired by [113, 75].

5.1. Model Checking for Small Natural Strategies

We begin by looking at NatATL$_r$ model checking under the assumption that the complexity bounds k used in formulas are constant or bounded. In other words, they are not a parameter of the model checking problem. Under this restriction, one can show a polynomial reduction to the model checking problem for CTL formulas. In consequence, we obtain the following result.

Theorem 5. *The model checking problem for* NatATL$_r$ *with fixed k is in* $\mathbf{P}$ *with respect to the size of the model and the length of the formula.*

Proof. First, consider the formula $\varphi = \langle\langle A\rangle\rangle^{\leq k}\gamma$, in which $A \subseteq \mathbb{A}\mathrm{gt}$ 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 $compl_\Sigma(s_A) \leq k$. Recall that Boolean formulas are combinations of atomic propositions from the set $Prop$ and Boolean connectives from $Bool$. Thus, there are $O((|Prop| + |Bool|)^k \cdot |Act|)$ possible guarded actions of length at most k, and hence $O(((|Prop| + |Bool|)^k \cdot |Act|)^k) = O((|Prop| + |Bool|)^{k^2}(|Act|)^k)$ possible strategies. The idea is to check them one by one, in an arbitrary order.

Given a collective strategy s_A, we can prune the CGS by removing all the edges that disagree with s_A. This operation costs $O(|t|)$ in the worst case, where t is the transition relation of the input CGS. So far we have dealt with the strategic operator in 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 [47], well-known to have complexity $O(|t| \cdot |\gamma|)$. The total complexity is thus $O((|Prop| + |Bool|)^{k^2}(|Act|)^k \cdot (|t| + (|t| \cdot |\gamma|))) = O((|Prop| + |Bool|)^{k^2}(|Act|)^k \cdot |t| \cdot |\gamma|)$, and hence polynomial in the size of the model and the length of the formula.

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 by solving the innermost subformula with 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 subformula. The procedure ends when we have dealt with the outermost strategic operator in the input formula. □

5.2. Model Checking: General Case

We now study the complexity for NatATL$_r$ with the bounds in strategic modalities given as variables. We start by showing $\mathbf{NP}$-completeness for formulas with a single strategic operator followed by a simple temporal subformula. Then, we adapt the proof to show $\mathbf{\Delta_2^P}$-completeness of model checking for the whole NatATL$_r$.

Proposition 6. *Model checking* 1NatATL$_r$ *is in* **NP** *with respect to the size of the model, the length of the formula, and the value of the bound* k*.*

Proof. Consider $\varphi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, in which $A \subseteq \mathbb{A}\mathrm{gt}$ and γ is a formula over Boolean connectives and atomic propositions. Now, enumerating all the suitable strategies and checking them one by one would be too expensive. To overcome this, we nondeterministically guess a collective strategy s_A, and proceed using the same reasoning as in the proof of Theorem 5. Since the size of s_A is polynomial in the size of the model, the complexity of the algorithm is **NP**. □

Remark 1. *We emphasize that the result is given with respect to the* value *of* k*, or equivalently the size of the* unary representation *of* k*. We will employ the same approach throughout the rest of the paper. The complexity of model checking, relative to the size of the binary representation of* k*, is left for future work.*

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 Proposition 7. In SAT, the main ingredients are a CNF formula $\varphi = C_1 \wedge \ldots \wedge C_n$ and m propositional variables from a set $X = \{x_1, \ldots, x_m\}$. Each clause C_i can be written as $C_i = x_1^{s(i,1)} \vee \ldots \vee x_m^{s(i,m)}$, 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 $x_1, \ldots, x_m$ such that φ holds. We construct the corresponding CGS M_φ as follows. There are two players: verifier $\mathbf{v}$ and refuter $\mathbf{r}$. The state space contains the 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 atomic propositions is $Prop = \{\mathsf{C_1}, \ldots, \mathsf{C_n}, \mathsf{x_1}, \ldots, \mathsf{x_m}, \mathsf{win}\}$. Furthermore, we label each clause/literal state 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 state q_0 to a clause state q_i. At q_i, verifier decides (by proceeding to a literal 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 agent $\mathbf{v}$ can use only 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 by the atomic proposition instead of states.

More formally, let $1 \leq i \leq n$ and $1 \leq j \leq m$. We define $M_\varphi = (\mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V)$, where:

- $\mathbb{A}\mathrm{gt} = \{\mathbf{v}, \mathbf{r}\}$,
- $St = \{q_0\} \cup St_{cl} \cup St_{prop} \cup \{q_\top\}$, where $St_{cl} = \{q_1, \ldots, q_n\}$, and $St_{prop} = \{q_{1,1}, \ldots, q_{1,m}, \ldots, q_{n,1}, \ldots, q_{n,m}\}$;

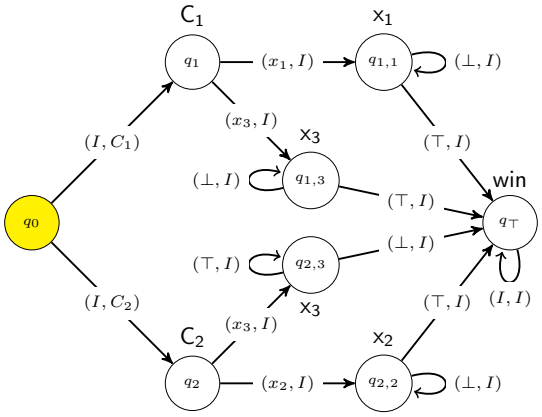

Figure 4: A CGS for checking satisfiability of $\varphi = (x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$. Action I denotes "idle." For simplicity, we omit the states that have no incoming edges.

- $Act = \{I, C_1, \ldots, C_n, x_1, \ldots, x_m, \top, \bot\}$,
- $d(\mathbf{v}, q_0) = d(\mathbf{v}, q_\top) = \{I\}$, $d(\mathbf{v}, q_i) = \{x_j \mid x_j \text{ or } \neg x_j \text{ is in } C_i\}$, $d(\mathbf{v}, q_{i,j}) = \{\top, \bot\}$; $d(\mathbf{r}, q_0) = \{C_1, \ldots, C_n\}$ 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;
- $Prop = \{\mathsf{C}_1 \ldots \mathsf{C}_n, \mathsf{x}_1, \ldots, \mathsf{x}_m, win\}$;
- $V(q_0) = \emptyset$, $V(q_i) = \mathsf{C_i}$, $V(q_{i,j}) = \mathsf{x_j}$, and $V(q_\top) = \mathsf{win}$;

As an example, model M_φ for $\varphi = (x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$ is presented in Figure 4.

Proposition 7. *$SAT(n, m, \varphi)$ iff $M_\varphi, q_0 \models \langle\langle \mathbf{v} \rangle\rangle^{\leq n+m} F\mathsf{win}$.*

Proof. **(⇒)** Firstly, if there is a valuation υ 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: (1) for each clause C_i we define a guarded action $(\mathsf{C_i},\ \alpha)$, where α is the action to go at the state literal that satisfy C_i in accordance with υ; and (2) for each literal x_j we define a guarded action $(\mathsf{x_j},\ \alpha)$, where α is the action to go in $q_\top$ in accordance with υ.

(⇐) 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 Propositions 6 and 7, the following result holds.

Theorem 8. *Model checking* 1NatATL$_r$ *is* **NP***-complete with respect to the size of the model, the length of the formula, and the value of the maximal bound k in the formula.*

To establish the model checking complexity for all formulas of NatATL$_r$, we adapt the above proofs in a similar way to [75]. Since the construction is rather technical, we present it in the appendix.

Theorem 9. *Model checking* NatATL$_r$ *is* $\mathbf{\Delta_2^P}$*-complete with respect to the size of the model, the length of the formula, and the maximal bound* k *in the formula.*

Proof. See Appendix A. □

6. Natural Abilities of Agents with Memory

Agents with memory can base their decisions on the history of the game, i.e., the sequence of states that has occurred so far. How can we represent conditions on such sequences? One possibility is to use states in some kind of automaton [124]. Here, we suggest that it is more intuitive for humans to represent conditions on histories by regular expressions over propositional formulas.

6.1. Natural Strategies with 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 *nR*-strategy) s_a for agent a is a sequence of appropriate pairs from $Reg(\mathcal{B}(Prop)) \times Act$. That is, it consists of pairs $(r,\ \alpha)$ where r is a regular expression over $\mathcal{B}(Prop)$, 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 *nr*-strategies, the last pair on the list is assumed to be simply $(\top^*,\ idle)$. The set of such strategies is denoted by Σ_a^{nR}. Finally, $match(\lambda[0,i], s_a)$ is the smallest $n \le length(s_a)$ 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^{nR}. Again, $out(q, s_A)$ returns the set of all paths from q, consistent with 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 of $out(q, s_A)$ that we gave in Section 4.1 for memoryless strategies.

The metrics from Section 4.1 extend to strategies with recall and collective strategies with recall in the straightforward way. Additionally, we define a variant of the metric $compl_\Sigma(\cdot)$ that skips the initial $\top^*$ whenever it appears in a regular expression:

Total size of the significant pattern: $compl_{\Sigma^*}(s_a) = \sum_{(r,\alpha)\in s_a} ||r||$, with $||\top^* \cdot r|| = ||r||$, $||\top^*|| = 1$, and $||r|| = |r|$ otherwise.

From now on, we will focus on the last metric for the complexity of strategies with recall. That is, unless explicitly specified, we will assume $compl(s_a) = compl_{\Sigma^*}(s_a)$.

Example 10 (Wild West explorer). *Consider the following nR-strategy* s *for a Wild West explorer:*

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

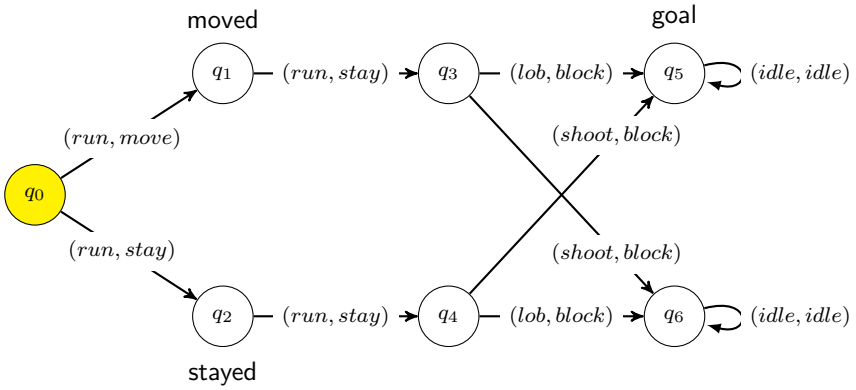

Figure 5: M_{RCup}: a model of robot soccer

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 he has a gun. If it doesn't work (item (4)), the agent should hide. Otherwise (item (5)), he waits and does nothing. For the complexity, we have that* $compl_{\#}(s) = 2$, $compl_{\max}(s) = 8$, $compl_{\Sigma}(s) = 27$, *and* $compl_{\Sigma^*}(s) = 23$.

Example 11 (RoboCup). *A very simple model of a soccer scenario is depicted in Figure 5. Robot 1 is running towards the goal with the ball. The goalkeeper (robot 2) can either stay close to the goal line, move towards the attacker. Then, after one more step, the attacker can either shoot straight or lob the ball over the goalkeeper. An nR-strategy for the attacker to score the goal is presented below:*

1. $(\top,\ run)$;
2. $(\top \cdot \top,\ run)$;
3. $(\top^* \cdot \mathsf{moved} \cdot \top,\ lob)$;
4. $(\top^* \cdot \mathsf{stayed} \cdot \top,\ shoot)$;
5. $(\top^*,\ idle)$.

The complexity of the strategy is $compl_{\Sigma^*}(s) = 11$.

Remark 2. *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 [124]. We will look closer at the connection in Section 6.4.*

6.2. 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 $\models_{nR}$ by replacing "$\models_{nr}$" with "$\models_{nR}$" and Σ_A^{nr} with Σ_A^{nR} in the clauses from Section 4.3, so that the clauses for strategic modalities become as follows:

- $M, q \models_{nR} \langle\langle A\rangle\rangle^{\leq k}\mathsf{X}\,\varphi$ iff there is a strategy $s_A \in \Sigma_A^{nR}$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[1] \models_{nR} \varphi$;
- $M, q \models_{nR} \langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{U}\,\psi$ iff there is a strategy $s_A \in \Sigma_A^{nR}$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have $M, \lambda[i] \models_{nR} \psi$ for some $i \geq 0$ and $M, \lambda[j] \models_{nR} \varphi$ for all $0 \leq j < i$.
- $M, q \models_{nR} \langle\langle A\rangle\rangle^{\leq k}\varphi\,\mathsf{W}\,\psi$ iff there is a strategy $s_A \in \Sigma_A^{nR}$ such that $compl(s_A) \leq k$ and, for each path $\lambda \in out(q, s_A)$, we have either that $M, \lambda[i] \models_{nR} \psi$ for some $i \geq 0$ and $M, \lambda[j] \models_{nR} \varphi$ for all $0 \leq j < i$, or that $M, \lambda[i] \models_{nR} \varphi$ for all $i \geq 0$.

We will refer to the logical system (NatATL, $\models_{nR}$) as NatATL$_R$.

Example 12 (RoboCup). *For the soccer model in Figure 5, we have* $M_{RCup}, q_0 \models_{nR} \langle\langle 1\rangle\rangle^{\leq 11}\mathsf{F}\,\mathsf{goal}$. *A natural strategy with recall that proves this was shown in Example 11.*

We note that the properties of NatATL$_r$, presented in Section 4.4, have their straightforward adaptations for natural strategies with recall, and they are proved in the analogous way. Thus, the CTL path quantifiers can be embedded in NatATL$_R$, and the fixpoint equivalences in general do not hold. Interestingly, the relationship between memoryless natural strategies and natural strategies with recall is also more complicated than in standard ATL. We look closer at the issue in the next subsection.

6.3. Relation between Memoryless and Memoryful Semantics of NatATL

It is well known that the semantics of ATL based on memoryless and perfect recall strategies coincide for agents with perfect information. This follows from the correctness of the model checking algorithm in [12], cf. also [113]. More 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 for natural strategies.

Theorem 10. *The following results hold in* NatATL*:*

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

Proof. **(1)** Given an *nr*-strategy s, it is possible to construct an *nR*-strategy s' that has the same behavior as 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 RoboCup CGS in Figure 5, and the strategy presented in Example 11. Clearly, the strategy shows that $M_{RCup}, q_0 \models_{nR} \langle\langle 1\rangle\rangle^{\leq 11}\mathsf{F}\,\mathsf{goal}$. On the other hand, $M_{RCup}, q_0 \not\models_{nr} \langle\langle 1\rangle\rangle^{\leq 11}\mathsf{F}\,\mathsf{goal}$. In fact, the formula is false for any bound k.

To see that, recall that conditions in natural memoryless strategies can only refer to Boolean properties of the current state. So, if there are states with the same valuations of atomic propositions, then each Boolean condition must have the same truth value in both states. Therefore, in M_{RCup}, it is impossible to define two different behaviors in states q_3 and q_4 within a natural memoryless strategy. In consequence, player 1 has no natural memoryless strategy to enforce reaching a state labeled with goal. □

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 overcome the limitation of natural memoryless strategies, we define a subclass of models, that we call *fully distinguishing models*. The idea behind this kind of models is the one used in [12, 84] to define the distinguishing models. The formal definition follows.

Definition 2. *Given a CGS M, we say that M is* fully distinguishing *iff, for all $S \subseteq St$, there exists $\mathsf{p} \in Prop$ such that $M, q \models \mathsf{p}$ iff $q \in S$.*

Theorem 11. *For every fully distinguishing model M, state q, subset of agents A, and formula $\varphi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, it holds that: $M, q \models_{nr} \varphi$ iff $M, q \models_{nR} \varphi$.*

Proof. **(⇒)** By Theorem 10.1.

(⇐) Assume now that $M, q \models_{nR} \varphi$. By definition, there is a strategy $s_A \in \Sigma_A^{nR}$ such that $compl(s_A) \leq k$, and for each path $\lambda \in out(q, s_A)$, we have $M, \lambda \models_{nR} \gamma$. From s_A, let us construct a memoryless strategy $s'_A \in \Sigma_A^{nR}$ such that the following facts hold: (1) $\forall \lambda \in out(q, s'_A)$, we have $M, \lambda \models_{nr} \gamma$ and (2) $compl(s'_A) \leq compl(s_A)$. We start at 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 (1) and (2) hold. Item (1) can be proved by induction. For simplicity, we omit the details. Item (2) 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 $compl(s'_A) \leq compl(s_A)$. □

6.4. Correspondence to Deterministic Finite-State Transducers

In [124], another useful representation of finite-memory strategies was introduced by means of finite input-output automata, also known as finite-state transducers.

Definition 3 (DFST [69, 124]). *A* deterministic finite-state transducer (DFST) *is a tuple* $\mathfrak{T} = (\mathfrak{S}, \mathfrak{s}_0, \mathfrak{In}, \mathfrak{Out}, \mathfrak{F}_{in}, \mathfrak{F}_{out})$*, where* $\mathfrak{S}$ *is a finite non-empty set of states of the transducer,* $\mathfrak{s}_0$ *is its initial state,* $\mathfrak{In}$ *is the input alphabet,* $\mathfrak{Out}$ *is the output alphabet,* $\mathfrak{F}_{in} : \mathfrak{S} \times \mathfrak{In} \to \mathfrak{S}$ *is the transition function of the transducer, and* $\mathfrak{F}_{out} : \mathfrak{S} \times \mathfrak{In} \to \mathfrak{Out}$ *is the output function.*

The idea is to have $\mathfrak{S}$ represent the possible states of agent a's internal memory that he uses when executing his strategy. The initial state corresponds to the initial memory value. In [124], the input and output symbols are the states of the CGS and the actions of a, respectively. With every step of the CGS, the transducer reads the current state of the system, and determines the next action of a to be executed, as well as the next memory state of the agent. Formally, let $\mathfrak{F}^+_{out} : \mathfrak{S} \times \mathfrak{In}^+ \to \mathfrak{Out}$ be the extension of the output function to sequences of inputs, defined recursively as follows: $\mathfrak{F}^+_{out}(\mathfrak{s}, (\mathfrak{i})) = \mathfrak{F}_{out}(\mathfrak{s}, \mathfrak{i})$, and $\mathfrak{F}^+_{out}(\mathfrak{s}, (\mathfrak{i}_1, \ldots, \mathfrak{i}_n)) = \mathfrak{F}^+_{out}(\mathfrak{F}_{in}(\mathfrak{s}, \mathfrak{i}_1), (\mathfrak{i}_2, \ldots, \mathfrak{i}_n))$. A strategy $s : H \to Act$ is a finite-memory strategy in M if there exists a DFST $\mathfrak{T}$ with $\mathfrak{In} = St_M$ and $\mathfrak{Out} = Act_M$ such that, for every history $h \in H$ in M, we have $s(h) = \mathfrak{F}^+_{out}(\mathfrak{s}_0, h)$. In that case, we also say that $\mathfrak{T}$ *implements* s.

In our approach, we assume that the agents "see" the world through the Boolean properties of states that can be formulated. If we want to compare natural strategies to the automata-based representation, we need to apply the same treatment.

Definition 4 (Natural DFST). *Let* $M = (\mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V)$ *be a concurrent game structure. A* natural DFST *for* M *is a DFST* $\mathfrak{T} = (\mathfrak{S}, \mathfrak{s}_0, \mathfrak{In}, \mathfrak{Out}, \mathfrak{F}_{in}, \mathfrak{F}_{out})$ *with* $\mathfrak{In} = 2^{Prop}$ *and* $\mathfrak{Out} = Act$*. That is, the transducer "reads" the atomic propositions satisfied in the current state, and outputs the next action of the agent.*

$\mathfrak{T}$ *defines the agent's strategy as follows: Let* $V^+ : H \to 2^{Prop}$ *be the pointwise extension of the valuation function to histories, i.e.,* $V^+(q_1 \ldots q_n) = V(q_1) \ldots V(q_n)$*. Now,* $s(h) = \mathfrak{F}^+_{out}(\mathfrak{s}_0, V^+(h))$*.*

It turns out that finite-memory strategies based on regular expressions and finite automata are equally expressive. Moreover, some natural strategies with recall are more succinct than DFST's.

Theorem 12. *Let* M *be a concurrent game structure, and* a *an agent in* M*. For every memoryful strategy* $s_a : H \to Act$ *of* a*, we have that* s_a *can be implemented by a natural strategy with recall iff it can be implemented by a natural DFST.*

Proof. **(⇒)** First, we show that from a natural strategy with recall s one can construct a DFST $\mathfrak{T}$ implementing the same strategy as s. Remember that $s = ((r_1,\ \alpha_1), \ldots, (r_n,\ \alpha_n))$, where every r_i is a regular expression over $\mathcal{B}(Prop)$. Thus, for each r_i, there exists a deterministic finite automaton D_i over 2^{Prop}, accepting the same language. One can produce it, e.g., by the following procedure:

1. Take r_i, build an equivalent NFA using the classical polynomial construction, and determinize it by means of the powerset construction [104];

2. Let $sat(\varphi) = \{\mathcal{X} \subseteq Prop \mid \mathcal{X} \models \varphi\}$ be the set of propositional valuations that satisfy $\varphi \in \mathcal{B}(Prop)$. For every transition $\mathfrak{s} \xrightarrow{\varphi} \mathfrak{s}'$ in the DFA obtained in point 1, replace it with the transitions $\mathfrak{s} \xrightarrow{\mathcal{X}} \mathfrak{s}'$, one per $\mathcal{X} \in sat(\varphi)$.

Now, let $D = D_1 \times \cdots \times D_n$ be the product automaton of all $D_1, \ldots, D_n$. We extend it to the transducer $\mathfrak{T}$ by adding the output function $\mathfrak{F}_{out}((\mathfrak{s}_1, \ldots, \mathfrak{s}_n), \mathcal{X}) = \alpha_i$ where i is the smallest number such that $(\mathfrak{s}'_1, \ldots, \mathfrak{s}'_n) = \mathfrak{F}_{in}((\mathfrak{s}_1, \ldots, \mathfrak{s}_n), \mathcal{X})$, and $\mathfrak{s}'_i$ is an accepting state in D_i.

(⇐) Secondly, we show how, from a natural DFST $\mathfrak{T}$, one can construct a natural strategy with recall s implementing the same strategy as $\mathfrak{T}$:

1. Take any arbitrary ordering $\mathfrak{s}_1, \ldots, \mathfrak{s}_n$ of states in $\mathfrak{T}$, and let D_i be the deterministic finite automaton obtained from $\mathfrak{T}$ by setting $\mathfrak{s}_i$ as the sole accepting state (and, of course, removing the output function). Notice that the languages accepted by $D_1, \ldots, D_n$ are pairwise disjoint. To see that, suppose that some sequence $\rho \in L(D_i) \cap L(D_j)$ for some $i \neq j$. But this means that the execution of ρ in $\mathfrak{T}$ can end up in both $\mathfrak{s}_i$ and $\mathfrak{s}_j$, which is impossible, as $\mathfrak{T}$ is deterministic;

2. For every i, take r_i to be a regular expression accepting the same language as D_i (it always exists!);

3. Obtain $\hat{r}_i$ from r_i by changing the alphabet from 2^{Prop} to $\mathcal{B}(Prop)$, and replacing each $\mathcal{X}$ in r_i by $conj(\mathcal{X}) = \bigwedge_{\mathsf{p} \in \mathcal{X}} \mathsf{p} \wedge \bigwedge_{\mathsf{p} \notin \mathcal{X}} \neg \mathsf{p}$, i.e., by the conjunction of all the literals satisfied by $\mathcal{X}$;

4. For every $\mathcal{X} \in 2^{Prop}$, let $\hat{r}_i^{\mathcal{X}} = \hat{r}_i \cdot conj(\mathcal{X})$. Note that the languages accepted by different $\hat{r}_i^{\mathcal{X}}$ are pairwise disjoint. To see this, suppose that some sequence $\rho \in L(\hat{r}_i^{\mathcal{X}}) \cap L(\hat{r}_j^{\mathcal{Y}})$ for some $i \neq j$ or $\mathcal{X} \neq \mathcal{Y}$. However, $\rho \in L(\hat{r}_i \cdot conj(\mathcal{X})) \cap L(\hat{r}_j \cdot conj(\mathcal{Y}))$ implies that $\mathcal{X} = \mathcal{Y}$ and $\rho[0, (|\rho| - 1)] \in L(\hat{r}_i) \cap L(\hat{r}_j)$. Thus, $L(D_i)$ and $L(D_j)$ are not disjoint, which is only possible for $i = j$;

5. Finally, let $\alpha_i^{\mathcal{X}} = \mathfrak{F}_{out}(\mathfrak{s}_i, \mathcal{X})$. We can now take s_a to be any ordering of pairs $(\hat{r}_i^{\mathcal{X}}, \alpha_i^{\mathcal{X}})$, augmented with $(\top, idle)$. Since the languages of $\hat{r}_i^{\mathcal{X}}$ are disjoint, the ordering of the guarded actions in s_a is irrelevant.

□

Note that the transformations of natural strategies to DFST in the above proof involved exponential blowup in the size of the representations. We will now show that in some cases it is unavoidable.

Lemma 13 ([60, Theorem 11]). *For infinitely many n, there are regular expressions r_n of alphabetic width[5] n over a binary alphabet, such that the minimal DFA accepting $L(r_n)$ has at least $\frac{5}{4}2^{\frac{n}{2}}$ states.*

[5] I.e., the length of r_n without parentheses and operators.

Theorem 14. *Natural strategies with recall can be exponentially more succinct than natural DFST's. More precisely, there exists a CGS M with agent a, and a sequence of natural strategies with recall s_n for a in M, such that all the implementations of s_n by natural DFST's are at least exponentially larger than s_n.*

Proof. We use the single-agent CGS M with two states $St = \{q_1, q_2\}$, two actions $Act = \{idle, move\}$ available everywhere, and transitions such that $idle$ never changes the state, and $move$ changes the state for the other. The sole atomic proposition p holds in q_1 but not in q_2.

For $m \geq 1$, take the m-th number n_m and a regular expression r_m that satisfy Lemma 13. Since the alphabetic width of r_m is n_m, we know that the length of r_m (counting also the operators) is at most $2n_m$. Let $s_m = ((r_m, move), (\top, idle))$. Suppose now that there exists a transducer $\mathfrak{T}_m$ over alphabet $2^{\{\mathsf{p}\}}$ with less than exponentially many states wrt $|r_m| = 2n_m$, implementing the same strategy. From $\mathfrak{T}_m$, we first construct an equivalent transducer $\mathfrak{T}'_m$ where all the incoming transitions to the same state have the same input (we simply create two copies of each state $\mathfrak{s}$, one for incoming transitions labeled by $\{\mathsf{p}\}$, and one for transitions labeled by $\emptyset$). Now, we transform $\mathfrak{T}'_m$ to a DFA D'_m by removing the output function and setting $\mathfrak{s}'$ as accepting whenever there is a sequence accepted by r_m that ends in $\mathfrak{s}'$. Note that, by construction, if there is a sequence in $L(r_m)$ ending in $\mathfrak{s}'$, then all the sequences ending in $\mathfrak{s}'$ are in $L(r_m)$. In consequence, $L(D'_m) = L(r_m)$. But D'_m has only twice as many states as $\mathfrak{T}_m$, which is still less than exponentially many wrt $|r_m|$. Thus, we have obtained a contradiction.

We note in passing that, even if the DFST corresponding to a natural strategy has polynomially many states with respect to the size of the strategy, it may have exponentially more transitions, as the shift between alphabets essentially requires conversion of the Boolean conditions φ to their Disjunctive Normal Form, and that often produces exponentially many disjuncts. □

We conjecture that the succinctness of natural strategies and that of natural DFST's are in fact incomparable. That is, we suspect that there are also scalable classes of DFST-implemented strategies, for whom all the equivalent natural strategies with recall are of at least exponential size. This is suggested by results such as [57, Theorem 3.4] and [60, Theorem 24]. However, proving it would be nontrivial, and is beyond the scope of this paper.[6]

7. Model Checking for Natural Strategies with Recall

In this section we investigate the model checking problem for NatATL with *nR*-strategies, i.e., NatATL$_R$. We consider both the case in which the bound of the strategies is a constant and the case when it is a variable. The following lemma allows us to use bounded tree unfoldings for the outcome sets of strategies with recall.

[6] The proof would require an in-depth analysis of the comparative succinctness of DFA's versus *regular expressions with complement*, because natural strategies with recall can easily simulate complementation. In general, regular expressions with complement can be even double-exponentially more succinct than ordinary regular expressions [57, Theorem 4.2].

Algorithm $mCheck^{const}_{\text{NatATL}_R}(M, q, \langle\langle A\rangle\rangle^{\leq k}\gamma)$:

```
1  for every s_A with compl(s_A) ≤ k do
2    if not IsLosing(s_A, M, q, p1 U p2) then return(true);
3  return (false);
```

Figure 6: Model checking NatATL$_R$ for simple goals, i.e., $\gamma \equiv \mathsf{p}_1 \mathsf{U} \mathsf{p}_2$ or $\gamma \equiv \mathsf{p}_1 \mathsf{W} \mathsf{p}_2$. The value of k is bounded by a constant

Lemma 15. *Let M be a CGS, $s_A = (s_{a_1}, \ldots, s_{a_n}) \in \Sigma_A^{nR}$ a natural strategy with recall of size $k = compl(s_A)$, and $\gamma \equiv \mathsf{p}_1 \mathsf{U} \mathsf{p}_2$ or $\mathsf{p}_1 \mathsf{W} \mathsf{p}_2$ a simple objective. In order to check if s_A enforces γ from $q \in St_M$, it is enough to consider the prefixes of length $|St_M| \cdot 2^{2k^2}$ of paths in $out(q, s_A)$.*

Proof. Consider the tree of outcome paths of s_A in M, starting from q. It can be obtained by an infinite process, based on the following notion of configuration: $C = (q_M, q_{reg_1}, \ldots, q_{reg_n})$ where q_M is the current state of M, and every q_{reg_i} is the current state of a deterministic finite automaton (DFA) accepting the ith regular expression in s_A. The initial configuration C_0 consists of q and the initial states of the DFA's.

Let C be the current configuration. The process takes, for each agent $a \in A$, the first DFA for a regular expression in s_a that is currently in an accepting state, and selects the corresponding action in s_a for execution by a. Then, for every possible tuple of responses from $\mathbb{A}\mathrm{gt} \setminus A$, a transition is added, leading to the configuration C' consisting of the successor state q' in M and the states of the DFA's updated accordingly. Note that, whenever the process revisits a previously encountered configuration, exactly the same transitions as before are added. Thus, whatever reachability objective $\gamma \equiv \mathsf{p}_1 \mathsf{U} \mathsf{p}_2$ can be validated (resp. safety objective $\gamma \equiv \mathsf{p}_1 \mathsf{W} \mathsf{p}_2$ invalidated), it can be done on the initial, cycle-free segment of the tree.

Finally, observe that s_A contains at most k regular expressions, and each expression is of length at most k. For every regular expression of length ℓ, there exists an equivalent nondeterministic finite automaton (NFA) with at most 2ℓ states (Thompson's construction). Finally, for every NFA with n states, there exists an equivalent DFA with at most 2^n states (powerset construction). Thus, the number of configurations is at most $|St_M| \cdot (2^{2k})^k = |St_M| \cdot 2^{2k^2}$. □

7.1. Model Checking for Small Strategies

When the bound of the strategies is fixed or bounded by a constant, we can reduce model checking to checking strategies one by one. For each strategy, we try to guess a path that invalidates it, verify the objective on the path, and revert the output. This leads to the following result.

Theorem 16. *The model checking problem for* NatATL$_R$ *with fixed or bounded k is in $\mathbf{\Delta_2^P}$ with respect to the size of the model and the length of the formula.*

Proof. Assume for the moment that we have a NatATL$_R$ formula $\varphi = \langle\langle A\rangle\rangle^{\leq k}\gamma$, where $A \subseteq \mathbb{A}\mathrm{gt}$ and γ is a formula with no nested strategic modalities. As for NatATL$_r$,

Algorithm $IsLosing(s_A, M, q, \mathsf{p}_1 \mathsf{U} \mathsf{p}_2)$:

```
1  count := 0; state := q;
2  size := compl(s_A); limit := 2^{2·size^2};
3  for every regular expression r_i ∈ s_A do
4    initialize the NFA A_i for r_i;
5  repeat
6    if M, state ⊨ p_2 then return(false);
7    if M, state ⊭ p_1 then return(true);
8    for each a ∈ A do
9      match := minimal i such that r_i ∈ s_a
10       and A_i is in an accepting state;
11     α_a := act_match(s_a);
12   for each a ∉ A do
13     nondeterministically choose α_a ∈ d(a, state);
14   state := t(state, α_1, ..., α_|Agt|);
15   for every NFA A_i do
16     nondeterministically update the state of A_i;
17   count := count + 1;
18 until count > limit;
19 return(true);
```

Figure 7: Oracle that tries to invalidate strategy s_A for a simple reachability goal $\mathsf{p}_1 \mathsf{U} \mathsf{p}_2$

we know that the collective strategy s_A that can be assigned to A is bounded, and we have that $compl_\Sigma(s_A) \leq k$. The main difference between *nr*-strategies and *nR*-strategies is in 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 combinations of atomic propositions ($Prop$), Boolean connectives ($Bool$), and regular expression constructors (Con). Thus, in this case, we have $O((|Prop| + |Bool| + |Con|)^k \cdot |Act|)$ possible different guarded actions and $O(((|Prop| + |Bool| + |Con|)^k \cdot |Act|)^k) = O((|Prop| + |Bool| + |Con|)^{k^2}(|Act|)^k)$ possible strategies. We check the strategies one by one; for each strategy, we use an oracle $IsLosing$ that returns "true" if it manages to guess a path invalidating the goal γ, and "false" otherwise. The case of simple reachability goals is presented in Figure 7; for simple safety goals, the oracle is defined analogously. It proceeds by nondeterministically unfolding a path consistent with strategy s_A from state q on until it either fulfills the goal, invalidates it, or exceeds the limit determined in Lemma 15.

Note that it is not possible anymore to obtain the set of outcome paths of s_A by simple pruning of M. To deal with that, we need to consider an unwinding of M, and then prune the unwinding. Clearly, the standard tree unwinding of M is infinite, and thus we need to consider a bounded unwinding. Fortunately, we can use Lemma 15 to obtain a bounded subtree.

The complexity of the procedure is as follows. The oracle runs in $O(2^{2k^2}) + O(k) + O(|St_M| \cdot 2^{2k^2} \cdot (k|\mathbb{A}gt| + |t| + |St|))$ steps. Since k is a constant, this reduces to

Algorithm $mCheck_{\text{NatATL}_R}(M, q, \langle\langle A\rangle\rangle^{\leq k}\gamma)$:

1 guess a strategy $s_A \in \Sigma_A^{nR}$ with $compl(s_A) \leq k$;
2 **return** (**not** $IsLosing(s_A, M, q, \mathsf{p}_1 \,\mathsf{U}\, \mathsf{p}_2)$);

Figure 8: Model checking NatATL$_R$ for simple goals; k is a parameter of the problem

$O(|St_M| \cdot (|\mathbb{A}\text{gt}| + |t| + |St|))$. Thus, the oracle runs in nondeterministic polynomial time with respect to the size of the model. In consequence, the algorithm in Figure 6 runs in time $\mathbf{P^{NP}} = \mathbf{\Delta_2^P}$.

For nested strategic modalities, we proceed recursively, bottom-up, which yields the complexity of $\mathbf{P^{\Delta_2^P}} = \mathbf{\Delta_2^P}$ for the whole problem. □

7.2. Model Checking: General Case

We now study the model checking complexity for NatATL$_R$ in case the bound over the strategies is a parameter of the problem.

Theorem 17. *Model checking* NatATL$_R$ *is in* $\mathbf{PSPACE}$ *with respect to the size of the model, the length of the formula, and the maximal bound* k *in the formula.*

Proof. For variable k, the algorithm in Figures 6 and 7 clearly runs in exponential time. To avoid using exponential time, we slightly change the main procedure, see Figure 8. Now, the oracle uses only polynomial space. This is because all the NFA's have at most $2k$ states. Thus, by using binary representations of variables $count$ and $limit$, they must be allocated at most $2\log_2(2^{2k^2}) = 4k^2$ memory cells. Thus, the complexity of the algorithm in Figure 8 is $\mathbf{NP^{NPSPACE}} = \mathbf{NP^{PSPACE}} = \mathbf{PSPACE}$. For nested strategic modalities, we again proceed recursively, which results in the complexity of $\mathbf{P^{PSPACE}} = \mathbf{PSPACE}$. □

8. Natural Ability in Concurrent Games

In the previous sections, we introduced the concept of a natural strategy, and studied how it changes the reasoning about what agents and their groups can achieve in multi-agent interaction. We used concurrent game structures to model the dynamics of interaction between agents. It is well known that CGS generalize the *extensive form* as well as the *repeated normal form* of perfect information games.[7] In other words, CGS allow to model all kinds of transition-action structures that occur among standard game models (plus some more). In this and subsequent sections, we look at *games of natural ability* that can be played over such structures.

[7] See [125, 98, 114] for an introduction to game theory.

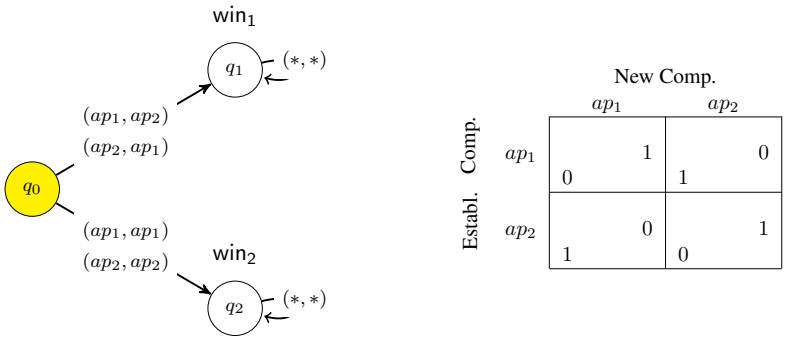

Figure 9: Simple market scenario: (a) concurrent game structure M_{market}; (b) normal form game

8.1. Concurrent Games

In game theory, a *game frame* defines the "arena" where players interact. The arena, together with a notion of a strategy, defines the space of strategic choices available to the players. This in turn determines the set of possible outcomes (in our case, paths) that can occur. A *game* is a frame plus a specification of players' preferences. In its most general version, agent a_i's preferences can be represented by a partial preorder over the possible outcomes in the game. A typical game-theoretic representation uses a payoff function (or utility function) that maps outcomes to real numbers; this way, one can represent any total preorder. For agents with qualitative objectives, ordinal preferences are often used, represented by a list of logical formulas $(\varphi_i^1, \ldots, \varphi_i^n)$. The idea is that the agent is happiest if φ_i^1 is achieved, otherwise would prefer to obtain φ_i^2, and so on [41, 62, 63]. In the simplest case of $n = 1$, this boils down to specification of the agent's *winning condition* φ_i, i.e., each agent gets a binary payoff: either win or lose [51, 24].

In this paper, we study concurrent games obtained by endowing the agents in a CGS with such binary winning conditions. We believe that our results generalize to concurrent games with finite ordinal preferences, but leave the broader case out to avoid overcomplicated presentation.

Formally, let M be a CGS with $|\mathbb{A}\mathrm{gt}| = n$. The objective of every agent $a \in \mathbb{A}\mathrm{gt}$ is defined by an LTL formula $\Phi(a)$ over $Prop$. Given a path λ in M, the payoff of the agent is 1 if $\lambda \models \Phi(a)$, or 0 otherwise. Given the intuition, the formal definition follows.

Definition 5. *A* concurrent game *is a tuple $G = (M, q_0, \Phi)$, where $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V\rangle$, is a concurrent game structure, $q_0 \in St$ is a state in M, and $\Phi : \mathbb{A}\mathrm{gt} \to \mathcal{L}_{LTL}$ is a mapping that assigns each agent with an LTL formula. We will often write Φ_a instead of $\Phi(a)$.*

Example 13 (Simple market scenario). *An established company (EC) and a new company (NC) have to choose the appearance for a product. Each company can choose between two different appearances for the product (ap_1 and ap_2). The established*

producer prefers the two products to look different (so that its customers will not be tempted to buy the newcomer's product), while the new company is better off when the products look alike. We can model the scenario by the concurrent game $G_{market} = (M_{market}, q_0, \Phi)$*, consisting of the CGS depicted in Figure 9a and objectives* $\Phi_{EC} = \mathrm{F}\mathsf{win}_1$*,* $\Phi_{NC} = \mathrm{F}\mathsf{win}_2$*.*

It is easy to see that the scenario has a similar structure to the well-known matching pennies *game, cf. Figure 9b.*

Example 14 (Ticket machine). *A concurrent game can be also constructed on top of the ticket machine CGS. For example,* $G_{ticket} = (M_{ticket}, q_0, \Phi)$ *with* $\Phi_c = \mathsf{F}\mathsf{ticket} \wedge \mathsf{G}\neg\mathsf{paid}$ *and* $\Phi_m = \neg\mathsf{GF}\mathsf{error}$ *expresses that the customer would like to get a ticket without ever paying, whereas the goal of the machine is to get to the error state at most finitely many times.*

In game theory, each agent wants to maximize his payoff, assuming some kind of rational behavior from the other players. In case of binary payoffs, i.e., winning conditions, the goal of a_i is to make Φ_i true. Rationality assumptions are typically formalized through so called *solution concepts*. Formally, a solution concept selects a subset of strategies (individual, coalitional, or strategy profiles), and designates them as rational. The best known game-theoretic solution concept is *Nash equilibrium*, whereas the one most studied in logical approaches to games is that of *surely winning strategy*. We will introduce both concepts, and define some corresponding decision problems, in the next subsections.

8.2. Decision Problems for Natural Ability in Games: Surely Winning

The most popular solution concept in logical approaches to strategic reasoning is *surely winning*. Player a wins if he has a strategy that obtains his goal no matter what the other players do. In game-theoretic terms, surely winning can be seen as an instance of *maxmin* for the special case of games with binary payoffs. The concept extends in a straightforward way to coalitions, see the formal definition below.

Definition 6. *Given a concurrent game* $G = (M, q_0, \Phi)$*, a subset of agents* $A \subseteq \mathbb{A}\mathrm{gt}$*, and a natural number* $k \in \mathbb{N}$*, we say that the natural collective strategy* s_A *of* A *is* surely winning *in* G *iff for all* $\lambda \in out(q_0, s_A)$ *and* $a \in A$ *it holds that* $\lambda \models \Phi_a$*. Moreover, coalition* A surely wins *in* G *under bound* k *iff it has a sure winning strategy of size at most* k*.*

SUREWIN is the decision problem that checks if a given coalition has a surely winning strategy.

Definition 7 (SUREWIN).
Input: *a concurrent game* G*, coalition* A*, and a natural number* k*;*
Output: true *if* A *surely wins in* G *under bound* k*, otherwise* false*.*

Note that the problem comes in fact in two variants, depending on our assumptions about the agents' memory. We can ask about *surely winning in natural memoryless strategies*, or *surely winning in natural strategies with recall*. To distinguish between

those, we will add the subscript *nr* or *nR* whenever the variant is not clear from the context. The same remark applies to all the other decision problems that we define and study for concurrent games.

Example 15 (Ticket machine). *Consider the concurrent game G_{ticket} in Example 14. It is easy to see that* SUREWIN$(G_{ticket}, \{c\}, k)$ = SUREWIN$(G_{ticket}, \{c, m\}, k)$ = *false regardless of the bound and the strategy type, because the only paths that obtain* ticket *must visit a state that satisfies* paid *before that. On the other hand,* SUREWIN$(G_{ticket}, \{m\}, k) = true$ *for any $k \geq 3$ in both memoryless and memoryful strategies (winning strategy: print if* paid*, reset if* ticket*, otherwise be helpful).*

We can also check if a particular strategy is surely winning. We call the problem "strategy checking," similar to *path checking* of LTL formulas [88].

Definition 8 (STRATEGYCHECKING).
Input: *a concurrent game G and a natural coalitional strategy s_A with bound k;*
Output: true *if s_A is a surely winning strategy, otherwise* false*.*

Note that deciding SUREWIN is closely related to model checking of NatATL. Like model checking, SUREWIN does not take into account the goals of the agents outside A. A classical solution concept that assumes rational behavior from all the agents is Nash equilibrium. We discuss it, and define the corresponding decision problems, in the next subsection.

8.3. Nash Equilibria in Natural Strategies

Nash equilibria are combinations of individual strategies that are stable under unilateral deviations. Formally, let us define *natural strategy profiles* as tuples $s_{\mathbb{A}\mathrm{gt}} = (s_1, \ldots, s_{|\mathbb{A}\mathrm{gt}|})$ of natural strategies, one per player. Note that every strategy profile $s_{\mathbb{A}\mathrm{gt}}$ determines exactly one infinite path in a given concurrent game G. We will denote the path by $path(s_{\mathbb{A}\mathrm{gt}})$. That is, $path(s_{\mathbb{A}\mathrm{gt}}) = \lambda$ such that $out(q_0, s_{\mathbb{A}\mathrm{gt}}) = \{\lambda\}$. Now, $s_{\mathbb{A}\mathrm{gt}}$ is a Nash equilibrium in G if each agent's part of $s_{\mathbb{A}\mathrm{gt}}$ is a best response to the collective strategy of the opponents.

Definition 9 (Best response). *Given a concurrent game $G = (M, q_0, \Phi)$, a player i, and a profile $s_{\mathbb{A}\mathrm{gt}} = (s_1, \ldots, s_i, \ldots, s_{|\mathbb{A}\mathrm{gt}|})$ of natural strategies under bound $k \in \mathbb{N}$, we say that s_i is a* best response *in $s_{\mathbb{A}\mathrm{gt}}$ under bound k iff $path(s_{\mathbb{A}\mathrm{gt}}) \not\models \Phi_i$ implies that, for all $s'_i \in \Sigma_i^{nr}$ such that $compl(s'_i) \leq k$, also $path((s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_{|\mathbb{A}\mathrm{gt}|})) \not\models \Phi_i$.*

Definition 10 (Nash equilibrium). *A natural strategy profile $s_{\mathbb{A}\mathrm{gt}} = (s_1, \ldots, s_{|\mathbb{A}\mathrm{gt}|})$ is a* Nash Equilibrium *in concurrent game G under bound k iff, for every $i \in \mathbb{A}\mathrm{gt}$, s_i is a best response in $s_{\mathbb{A}\mathrm{gt}}$ under bound k.*

Example 16 (Simple market scenario). *Take the simple market game in Example 13. It is easy to see, by looking at the payoff table in Figure 9b, that it has no Nash equilibria in natural memoryless strategies.*

Remark 3. *A more general version of Nash equilibrium in natural strategies could also be considered, with bounds $k_1, \ldots, k_{|\mathbb{A}gt|}$. In that case, each player would possibly have a different bound on the complexity of his strategies. We conjecture that the complexity results follow the same pattern as for the simple case of a single k, but do not pursue this line in detail here.*

We will now define the corresponding decision problems.

Definition 11 (IsNash).
Input: *a concurrent game G and a natural strategy profile $s_{\mathbb{A}gt}$ with bound k;*
Output: true *if $s_{\mathbb{A}gt}$ is a Nash equilibrium, otherwise* false.

Note that we take into account only pure – that is, non-randomized – strategies here. Probabilistic strategies would require a different treatment. In particular, we would have to redefine the models and the notion of a strategy more in the vein of probabilistic logics for Markov decision processes [83, 46, 50, 73]. This is certainly an interesting path, and we plan to investigate it in the future.

Since we only consider pure strategies, the existence of Nash equilibria is not guaranteed, as demonstrated by Example 16. In consequence, it makes sense to ask if a given game has at least one stable point.

Definition 12 (ExistNash).
Input: *a concurrent game G and a natural number k;*
Output: true *if there exists a strategy profile $s_{\mathbb{A}gt}$ which is a Nash equilibrium in G under bound k, otherwise* false.

Even more interestingly, we can ask whether a given player wins in some (or dually, all) Nash equilibria.

Definition 13 (WinsSomeNash).
Input: *a concurrent game G, player i, and a natural number k;*
Output: true *if there exists a strategy profile $s_{\mathbb{A}gt}$ which is a Nash equilibrium in G under bound k and $path(s_{\mathbb{A}gt}) \models \Phi_i$, otherwise* false.

Definition 14 (WinsAllNash).
Input: *a concurrent game G, player i, and a natural number k;*
Output: true *if $path(s_{\mathbb{A}gt}) \models \Phi_i$ for all strategy profiles $s_{\mathbb{A}gt}$ which are Nash equilibria in G under bound k, otherwise* false.

The above decision problems come very close to the recent work on *equilibrium checking* by Wooldridge, Gutierrez and colleagues [116, 128, 63]. The difference is that they look at arbitrary combinatorial strategies with perfect recall, whereas we consider natural strategies under bound k (both perfect recall and memoryless). We will see in Section 9 that this reduces the computational complexity: from $\mathbf{2EXPTIME}$ for combinatorial strategies [128, 63] to $\mathbf{EXPSPACE}$ for natural strategies with recall, and between $\mathbf{\Sigma_2^P}$ and $\mathbf{PSPACE}$ for natural memoryless strategies.

We also note that a similar set of decision problems can be defined for many other solution concepts, such as *undominated strategies*, *Stackelberg equilibrium*, *subgame-perfect Nash equilibrium*, and so on. We leave a proper exploration of the possibilities for future work.

Algorithm $SureWin(G, A, k)$:

1 $s_A = GuessStrat(G, A, k)$;
2 Prune M according to s_A, obtaining model M';
3 **return** $mCheck_{\text{CTL}^*}(M', q_0, \mathsf{A} \bigwedge_{i \in A} \Phi_i)$;

Figure 10: Algorithm to decide SUREWIN.

Algorithm $GuessStrat(G, A, k)$:

1 $size := 0$;
2 **for** every $i \in A$ do
3 $s_i := ()$;
4 repeat
5 Guess (β, α) with $compl(((\beta, \alpha))) \leq \max(1, k - size)$;
6 $size := size + compl(((\beta, \alpha)))$;
7 until $((\top, idle)$ **or** $size > k)$;
8 **if** $size > k$ then **return**(false);
9 **return** (s_A);

Figure 11: Algorithm to guess the natural collective strategy.

9. Complexity Results for Games of Natural Ability

We now study the asymptotic complexity of the decision problems introduced in the previous section. A table summarizing the results obtained here is presented at the end of the paper (Section 10, Figure 17).

9.1. Surely Winning in Memoryless Strategies

We start with the decision problem SUREWIN, which concerns checking the existence of a surely winning strategy. We show that SUREWIN is as difficult as LTL model checking, namely that it is **PSPACE**-complete.

Proposition 18. SUREWIN *is in* **PSPACE**.

Proof. Consider a concurrent game $G = (M, q_0, \Phi)$, a coalition of agents A, and a natural number bound k. In Figure 10 we give an algorithm that decides SUREWIN(G, A, k). The algorithm is nondeterministic (see line 1) and takes space polynomial in the input (due to procedure $mCheck_{\text{CTL}^*}$ called in line 3). The fact that **PSPACE** = **NPSPACE** concludes the proof. □

Proposition 19. SUREWIN *is* **PSPACE***-hard.*

Proof. To prove it, we use a reduction from LTL model checking, known to be **PSPACE**-complete [110]. The general idea is as follows: we take an arbitrary Kripke structure and transform it to a CGS with two players: the "real" agent a who fully controls the next transition of the system, and the "dummy" agent b who has no influence

on the choice of the next state. Then, the "dummy" can enforce those – and only those – path properties that hold on all the possible paths in the system.

Formally, given a Kripke structure K and an LTL formula φ, LTL model checking is the decision problem of verifying whether K meets φ, i.e., for all paths λ starting from an initial state of K it holds that $\lambda \models \varphi$. A Kripke structure over a set of atomic proposition AP is a tuple $K = (W, w_0, R, L)$, where $W = \{w_0, \ldots, w_n\}$ is the set of states, w_0 is the initial state, $R \subseteq W \times W$ is the transition relation, and $L : W \to 2^{AP}$ is a labeling function. Let us now reduce the LTL model checking problem to SUREWIN. Given K and φ as above, we construct a game $G = (M, q_0, \Phi)$ with $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V \rangle$, where:

- $\mathbb{A}\mathrm{gt} = \{a, b\}$;
- $St = W$;
- $Act = Act_a \cup Act_b$, where $Act_a = W$ and $Act_b = \{idle\}$;
- $d_a(w) = \{w' | R(w, w')\}$ and $d_b(w) = \{idle\}$, for each $w \in St$;
- $t(w, (w', idle)) = \begin{cases} w' & \text{if } w, w' \in St \text{ and } (w, w') \in R; \\ \emptyset & \text{otherwise.} \end{cases}$
- $Prop = AP \cup \{\mathsf{p_w} | w \in St\}$;
- $V(w) = L(w) \cup \{\mathsf{p_w}\}$, for each $w \in St$;
- $q_0 = w_0$;
- $\Phi = \{\Phi_a, \Phi_b\}$, such that $\Phi_a = \neg\varphi$ and $\Phi_b = \varphi$.

We now prove that the above reduction is correct by showing that $K \models \varphi$ iff SUREWIN$(G, b, 1) = true$.

(⇒) Assume that $K \models \varphi$. By definition, for each path λ in K it holds that $\lambda \models \varphi$. Consider now the game G constructed as above and the strategy $((\top,\ idle))$ for b. One can see that $compl(s_b) \leq 1$ and for all $\lambda \in out(q_0, s_a)$ it holds that $\lambda \models \Phi_b$. Hence, SUREWIN$(G, a, 1) = true$.

(⇐) Assume now SUREWIN$(G, b, 1) = true$. Then, by Definition 6 we know that there is a natural memoryless strategy s_a with $compl(s_a) \leq 1$ such that for all $\lambda \in out(q_0, s_a)$ it holds that $\lambda \models \Phi_b$. It is easy to see that the only b's strategy of size 1 is $((\top,\ idle))$. This means that $out(q_0, s_a)$ contains all the possible paths starting in q_0, and each of them satisfies the objective of player b. From this we directly derive that for each path λ in K it holds that $\lambda \models \varphi$. Hence, $K \models \varphi$. □

Putting together the results of Proposition 18 and Proposition 19, we obtain the following result.

Theorem 20. SUREWIN *is* **PSPACE***-complete with respect to the size of the game and the value of the bound k (even for coalitions of size 1).*

Somewhat surprisingly, checking whether a *given* strategy surely wins is not easier.

Theorem 21. STRATEGYCHECKING *is* **PSPACE***-complete with respect to the size of the game and the value of the bound k.*

Algorithm $IsNotNash(G, s_{\mathbb{A}\mathrm{gt}}, k)$:

```
1  for every i ∈ Agt do
2    if path(s_Agt) ⊭ Φ_i then
3      Guess s'_i with compl(s'_i) ≤ k
4      if path((s_1, ..., s_{i-1}, s'_i, s_{i+1}, ..., s_{|Agt|})) ⊨ Φ_i then return(true);
5  return (false);
```

Figure 12: Algorithm to decide the complement of IsNash.

Proof. The upper bound follows from the fact that the problem can be solved by the algorithm for $SureWin$ without the first step (i.e., without guessing the strategy). For the lower bound, one can use the same reduction as in the proof of Proposition 19, asking whether the "idle" strategy of the dummy player surely wins (instead of asking whether the dummy has a winning strategy). □

9.2. Nash Equilibria in Memoryless Strategies

Now we examine the complexity of the decision problems related to Nash equilibrium.

Proposition 22. IsNash *is in* **coNP**.

Proof. In Figure 12, we introduce an algorithm that uses an **NP** procedure to check whether a strategy profile $s_{\mathbb{A}\mathrm{gt}}$ is not a Nash Equilibrium. In details, starting with a strategy profile $s_{\mathbb{A}\mathrm{gt}}$ we check that for each player i, if the path generated by the strategy profile does not satisfy his own goal Φ_i, then no unilaterally deviation of the player i will allow him to satisfy his goal as well. Since **coNP** is the class of problems whose complements are in **NP**, the result immediately follows. □

Proposition 23. IsNash *is* **coNP**-*hard.*

Proof. We prove **NP**-hardness for the complement problem IsNotNash. To this end, we can use an approach similar to the one exploited in Theorem 8. Precisely, given a formula φ in SAT, we construct a game $G = (M_\varphi, q_0, \Phi)$, where M_φ is as in Theorem 8, q_0 is the initial state of M_φ and $\Phi = \varphi[\mathrm{F}\mathsf{p}_\mathsf{i}/\neg x_i, \mathrm{F}\bar{\mathsf{p}}_\mathsf{i}/x_i]$. By using an adaption of Proposition 7, it is easy to check that $\mathrm{SAT}(\varphi, m) = true$ iff IsNotNash$(G, \mathbf{v}, m)) = false$. In consequence, we get **coNP**-hardness for the complement problem, i.e., IsNash. □

By putting together Propositions 22 and 23, we obtain the following result.

Theorem 24. IsNash *is* **coNP**-*complete with respect to the size of the game and the value of the bound* k.

We now continue with WinsSomeNash.

Proposition 25. WinsSomeNash *is in* $\mathbf{\Sigma_2^P}$.

Algorithm $WinsSomeNash(G, i, k)$:

```
1  s_Agt = GuessStrat(G, Agt, k);
2  if path(s_Agt) ⊨ Φ_i and not IsNotNash(G, s_Agt, k) then return (true);
3  return (false);
```

Figure 13: Algorithm to decide WINSSOMENASH.

Proof. In Figure 13, we present an algorithm that uses an **NP** procedure to select the strategy profile $s_{\mathbb{A}\mathrm{gt}}$, calling another **NP** procedure (depicted in Figure 12) which checks whether $s_{\mathbb{A}\mathrm{gt}}$ is a Nash equilibrium. We recall that given a path λ and an LTL formula φ, checking whether λ satisfies φ requires polynomial time [88]. Therefore, the total complexity to decide WINSSOMENASH is $\mathbf{\Sigma_2^P}$. □

We will show a tight lower bound by a reduction from QBF_2, i.e., the satisfiability problem for Quantified Boolean Formulas with at most one alternation of quantifiers. First, we recall the definition of QBF and QBF_2. Then, in Proposition 27 we give the mentioned lower-bound.

Definition 15 ([115]). *A (fully) quantified Boolean formula is a formula in quantified propositional logic where every variable from a finite set is quantified by using either an existential or a universal quantifier. Formula* $\varphi = \exists x\, \forall y\, \forall z\, \exists w\, ((x \wedge \neg y) \vee (z \wedge y) \vee \neg w$ *is an example. A quantified Boolean formula is in* prenex normal form *if it has two basic parts: a portion containing only quantifiers and a portion containing an unquantified Boolean formula. By* QBF_i*, we denote the set of QBF formulas in prenex normal form with at most* i *types of quantifiers following each other (in other words, at most* $i-1$ *alternations of quantifiers). For instance,* φ *is a formula of* QBF_3*.*

Note that it suffices to consider only formulae in negation normal form (i.e., negations allowed only at the level of literals), as any other formula can be transformed to an equivalent one in linear time.

Lemma 26 ([115]). *For* QBF_i*, checking if formula* φ *holds is* $\mathbf{\Sigma_i^P}$*-complete with respect to the length of* φ*.*[8]

Proposition 27. WINSSOMENASH *is* $\mathbf{\Sigma_2^P}$*-hard.*

Proof. We adapt the reduction used in Section 5 to prove that $1\mathrm{NatATL}_r$ is **NP**-hard. Specifically, let $\varphi = \exists x_1 \ldots x_n \forall x_{n+1} \ldots x_m\ \psi$ be a QBF_2 formula in negation normal form. We build a game G with two players, the verifier (**v**) and the refuter (**r**), where **v** tries to show that φ is true, while **r** does the opposite. To this aim, we use states x_i, $1 \leq i \leq m$, covering all variables in φ, as points of choice for **v** and **r**. The verifier controls the existentially quantified variables, and the refuter the universally quantified ones. Moreover, states p_j and $\bar{p}_j$, $1 \leq i \leq m$, represent whether the player's choice for x_j has been 0 or 1. More formally, $G = (M, q_0, \Phi)$ with $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V \rangle$, where:

[8] $\mathbf{\Sigma_i^P}$ is defined recursively by: $\mathbf{\Sigma_1^P} = \mathbf{NP}$; $\mathbf{\Sigma_i^P} = \mathbf{NP}^{\mathbf{\Sigma_{i-1}^P}}$.

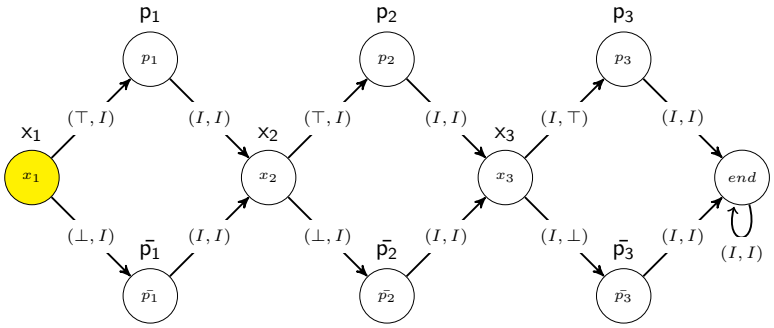

Figure 14: The concurrent game structure corresponding to the QBF$_2$ formula $\varphi = \exists x_1, x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$ obtained by the construction of Proposition 27. The objectives are: $\Phi_{\mathbf{v}} = (\mathrm{F}\mathsf{p}_1 \wedge \mathrm{F}\bar{\mathsf{p}}_2) \vee \mathrm{F}\mathsf{p}_3$, $\Phi_{\mathbf{r}} = \neg((\mathrm{F}\mathsf{p}_1 \wedge \mathrm{F}\bar{\mathsf{p}}_2) \vee \mathrm{F}\mathsf{p}_3)$.

- $\mathbb{A}\mathrm{gt} = \{\mathbf{v}, \mathbf{r}\}$;
- $St = \{x_1, \ldots, x_m\} \cup \{p_1, \ldots, p_m\} \cup \{\bar{p}_1, \ldots, \bar{p}_m\} \cup \{end\}$;
- $Act = \{\top, \bot, I\}$;
- $d_{\mathbf{v}}(s_i) = \begin{cases} \{\top, \bot\} & \text{if } s_i = x_i \text{ and } 1 \leq i \leq n; \\ \{I\} & \text{otherwise.} \end{cases}$

  $d_{\mathbf{r}}(s_i) = \begin{cases} \{\top, \bot\} & \text{if } s_i = x_i \text{ and } n+1 \leq i \leq m; \\ \{I\} & \text{otherwise.} \end{cases}$
- $t(x_i, (\top, I)) = p_i$, $t(x_i, (\bot, I)) = \bar{p}_i$, $t(x_j, (I, \top)) = p_j$, $t(x_j, (I, \bot)) = \bar{p}_j$, $t(s_k, (I, I)) = x_{k+1}$, $t(s, (I, I)) = end$, where $s_k \in \{p_k, \bar{p_k}\}$, $s \in \{p_m, \bar{p_m}, end\}$, $1 \leq i \leq n$, $n+1 \leq j \leq m$, and $1 \leq k < m$.
- $Prop = \{\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{m}\} \cup \{\mathsf{p}_1, \ldots, \mathsf{p}_\mathsf{m}\} \cup \{\bar{\mathsf{p}}_1, \ldots, \bar{\mathsf{p}}_\mathsf{m}\}$;
- $V(x_i) = \{\mathsf{x}_\mathsf{i}\}$, $V(p_i) = \{\mathsf{p}_\mathsf{i}\}$, $V(\bar{p}_i) = \{\bar{\mathsf{p}}_\mathsf{i}\}$, and $V(end) = \emptyset$;
- $q_0 = x_1$;
- $\Phi = \{\Phi_{\mathbf{v}}, \Phi_{\mathbf{r}}\}$, such that $\Phi_{\mathbf{v}} = tr(\varphi)$ and $\Phi_{\mathbf{r}} = \neg tr(\varphi)$, where $tr(\varphi) = \psi[\mathrm{F}\bar{\mathsf{p}}_\mathsf{i}/\neg x_i, \mathrm{F}\mathsf{p}_\mathsf{i}/x_i]$. That is, we remove the quantifiers from φ, replace every positive occurrence of x_i with $\mathrm{F}\mathsf{p}_\mathsf{i}$, and every negative occurrence of x_i with $\mathrm{F}\bar{\mathsf{p}}_\mathsf{i}$.

As an example, Figure 14 presents the game G for the QBF$_2$ formula $\varphi = \exists x_1 x_2\ \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$ that comes from our reduction.

We conclude by proving the correctness of our reduction. Formally, we prove that $\mathrm{QBF}_2(\varphi, n, m) = true$ iff WINSSOMENASH$(G, \mathbf{v}, max(n, m-n)) = true$.

(⇒) First, assume that $\mathrm{QBF}_2(\varphi, n, m) = true$. By definition, there exists an assignment χ for the variables $x_1, \ldots, x_n$ such that for all the assignments for the variables $x_{n+1}, \ldots, x_m$ the formula φ is true. Now, consider the game G obtained as described above. We first show that there is a surely winning for the verifier, i.e., a strategy $s_{\mathbf{v}}$ with $compl(s_{\mathbf{v}}) \leq n$ such that for all $\lambda \in out(q_0, s_{\mathbf{v}})$ it holds that $\lambda \models \Phi_{\mathbf{v}}$.

We construct $s_{\mathbf{v}}$ as follows: for each variable x_i, it contains a guarded action $(\mathsf{x_i},\ a)$, where $a = \top$ if x_i is true in the assignment χ, else $a = \bot$. Consequently, if $a = \top$ (resp., $a = \bot$) then every $\lambda \in out(q_0, s_{\mathbf{v}})$ eventually visits the state p_i (respectively $\bar{p_i}$) in which the atomic proposition $\mathsf{p_i}$ (respectively $\bar{\mathsf{p}}_\mathsf{i}$) holds.

Since, $s_{\mathbf{v}}$ is surely winning, it makes $\Phi_{\mathbf{v}}$ true (and hence $\Phi_{\mathbf{r}}$ false) for every strategy $s_{\mathbf{r}}$ of the refuter. Take any arbitrary $s_{\mathbf{r}}$ under bound $m-n$. Clearly, $s_{\mathbf{r}}$ is a best response to $s_{\mathbf{v}}$, and $s_{\mathbf{v}}$ is a best response to $s_{\mathbf{r}}$. Thus, $(s_{\mathbf{v}}, s_{\mathbf{r}})$ is a Nash equilibrium under bound $max(n, m-n)$. In consequence, WINSSOMENASH$(G, \mathbf{v}, max(n, m-n)) = true$.

(⇐) Conversely, assume that WINSSOMENASH$(G, \mathbf{v}, max(n, m-n)) = true$. Then, by Definition 13, there is a natural memoryless strategy profile $(s_{\mathbf{v}}, s_{\mathbf{r}})$ with $compl((s_{\mathbf{v}}, s_{\mathbf{r}})) \leq max(n, m-n)$ that is a Nash equilibrium and $path((s_{\mathbf{v}}, s_{\mathbf{r}})) \models \Phi_{\mathbf{v}}$. This also means that $path((s_{\mathbf{v}}, s_{\mathbf{r}})) \not\models \Phi_{\mathbf{r}}$, and hence $\Phi_{\mathbf{v}}$ holds on every $\lambda \in out(q_0, s_{\mathbf{v}})$. This in turn means that, for the assignment of variables x_i controlled by $\mathbf{v}$, formula ψ holds no matter what assignment is chosen by $\mathbf{r}$. Hence, QBF$_2(\varphi, n, m) = true$. □

By putting together Propositions 25 and 27, we obtain the following result.

Theorem 28. WINSSOMENASH *is* $\mathbf{\Sigma_2^P}$*-complete with respect to the size of the game and the value of the bound* k*.*

The complexity of EXISTNASH can be established by an adaptation of the techniques that we used for WINSSOMENASH. We state the theorem below; the detailed proofs are in Appendix B.

Theorem 29. EXISTNASH *is* $\mathbf{\Sigma_2^P}$*-complete with respect to the size of the game and the value of the bound* k*.*

Deciding if all the Nash equilibria satisfy a given property is in the dual complexity class.

Proposition 30. WINSALLNASH *is in* $\mathbf{\Pi_2^P}$.[9]

Proof. In Figure 15, we present an algorithm to check the complement problem of WINSALLNASH, i.e LOSESSOMENASH. The algorithm uses an $\mathbf{NP}$ procedure to select the strategy profile $s_{\mathbb{A}\mathrm{gt}}$ and calls another $\mathbf{NP}$ procedure (depicted in Figure 12) to check whether $s_{\mathbb{A}\mathrm{gt}}$ is a Nash equilibrium. So, LOSESSOMENASH is in $\mathbf{\Sigma_2^P}$ and thus WINSALLNASH is in $\mathbf{\Pi_2^P}$. □

Proposition 31. WINALLNASH *is* $\mathbf{\Pi_2^P}$*-hard.*

Proof. We can use a reduction from the complement problem of WINSSOMENASH. In particular, given a game $G = (M, q_0, \Phi)$, where $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V \rangle$, q_0 is the initial state, and Φ are the objectives we build a game $G^\star = (M, q_0, \Phi^\star)$, where $\Phi^\star$ is the complement function of Φ, i.e. for each agent i we have that $\Phi_i^\star = \neg\Phi_i$.

[9] $\mathbf{\Pi_2^P} = \mathbf{co{-}NP^{NP}}$ is the class of complement problems to $\mathbf{\Sigma_2^P}$.

Algorithm $LosesSomeNash(G, i, k)$:

```
1  s_Agt = GuessStrat(G, Agt, k);
2  if path(s_Agt) ⊭ Φ_i and not IsNotNash(G, s_Agt, k) then return (true);
3  return (false);
```

Figure 15: Algorithm to decide the complement of WINSALLNASH.

The proof of correctness directly follows from the fact that M is the same for both games. Therefore, the strategies produce the same outcomes. □

The following is a corollary.

Theorem 32. WINALLNASH *is* $\mathbf{\Pi_2^P}$*-complete with respect to the size of the game and the value of the bound* k*.*

9.3. Complexity Results for Agents with Memory

We conclude this section by studying all the decision problems presented in Section 8.2 under the assumption that players' strategies are memoryful. In most settings, strategies with memory make the related decision problems more computationally expensive. Here, we prove that the solution concepts for natural games with memory share the **EXPSPACE** upper bound. This comes by applying the following reasoning. Take all the problems that we have considered above and the procedures that we have used to check the satisfaction of the goals. A key step in all these procedures is to check whether a path satisfies the designed goal. In order to verify if all paths in a graph satisfy a given LTL formula, it suffices to check their prefixes of length $O(2^n)$, where n is the number of nodes in the graph [110]. Moreover, the unfolding graph of a given CGS for natural strategies with recall has $O(|St_M| \cdot 2^{2k^2})$ nodes (see the proof of Lemma 15). Thus, to verify if a strategy profile obtains the LTL objective, we need a counter that checks if a given bound of range $O(2^{|St_M| \cdot 2^{2k^2}})$ has been exceeded. This can be implemented with $O(|St_M| \cdot 2^{2k^2})$ memory cells, using a binary representation of the counter. In consequence, we obtain the **NEXPSPACE** upper bound for the problem. By Savitch's theorem, this is equivalent to **EXPSPACE**.

Theorem 33. *The decision problems* SUREWIN*,* STRATEGYCHECKING*,* ISNASH*,* WINSSOMENASH*,* EXISTNASH*, and* WINALLNASH *for natural strategies with recall are in* **EXPSPACE** *with respect to the size of the game and the value of the bound* k*.*

Moreover, for small bounds k, the complexity is much more optimistic.

Theorem 34. *If the bound* k *is constant, then* SUREWIN*,* STRATEGYCHECKING*,* ISNASH*,* WINSSOMENASH*,* EXISTNASH*, and* WINALLNASH *for natural strategies with recall are in* **PSPACE** *with respect to the size of the game.*

	memoryless	with 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 $\mathbf{PSPACE}$
NatATL, variable k	$\mathbf{\Delta_2^P}$-complete	in $\mathbf{PSPACE}$

Figure 16: Summary of complexity results for model checking NatATL

	memoryless	with recall
SUREWIN	$\mathbf{PSPACE}$-complete	in $\mathbf{EXPSPACE}$
STRATEGYCHECKING	$\mathbf{PSPACE}$-complete	
ISNASH	$\mathbf{coNP}$-complete	(in $\mathbf{PSPACE}$ for fixed k)
WINSSOMENASH	$\mathbf{\Sigma_2^P}$-complete	
EXISTNASH	$\mathbf{\Sigma_2^P}$-complete	
WINSALLNASH	$\mathbf{\Pi_2^P}$-complete	

Figure 17: Summary of complexity results for natural ability in concurrent games

10. Summary and Future Work

In this paper, we propose an alternative take on strategic reasoning, that allows to reason about agents who 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 used. This is formalized by NatATL, a new variant of alternating-time temporal logic where only such "natural" strategies are allowed as witnesses to formula $\langle\langle A\rangle\rangle^{\leq k}\gamma$. We argue that, in many cases, this is a more accurate view of ability than the one which admits any function from sequences of states to actions.

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, see the summary presented in Figure 16. Thus, reasoning about *small* natural memoryless strategies is no more difficult than for arbitrary ATL strategies (and in practice we expect it to be much 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 memoryful semantics do not coincide. Moreover, we show that our representation of memoryful strategies is equally expressive, and incomparably succinct to deterministic finite-state transducers from [124].

Having investigated the logical reasoning about natural strategies, we turn to the broader issue of natural strategic abilities in *concurrent games with* LTL*-definable winning conditions*. We study a number of decision problems based on *surely winning strategies* and *Nash equilibrium*; our complexity results are summarized in Figure 17. It turns out that, besides the conceptual advantage, reasoning about bounded natural strategies significantly decreases the complexity of "rational verification" for multi-agent systems. More precisely, the complexity goes down from $\mathbf{2EXPTIME}$

for arbitrary combinatorial strategies [128, 63] to **EXPSPACE** for natural strategies with recall, and between $\mathbf{\Sigma_2^P}$ and **PSPACE** for natural memoryless strategies.

The research reported here opens up many exciting paths for future work. Most importantly, we plan to extend the framework to natural strategies with imperfect information. We would also like to enhance our results on logical strategic reasoning 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. For natural ability in concurrent games, it seems worthwhile to consider decision problems based on other solution concepts (such as undominated strategies or Stackelberg equilibrium), and apply them to actual application domains, such as reasoning about coercion-resistance and usability of some e-voting platforms. We would also like to look at other natural representations 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 [32, 14].

Acknowledgements. We thank Valentin Goranko and Dilian Gurov for their comments regarding the relationship between natural strategies and deterministic finite transducers, and Sasha Rubin for the reference to robotics. Wojciech Jamroga acknowledges the support of the National Centre for Research and Development, Poland (NCBR), and the Luxembourg National Research Fund (FNR), under the PolLux/FNR-INTER project VoteVerif (POLLUX-IV/1/2016). The research was partly conducted during a research stay of Wojciech Jamroga at the Università degli Studi di Napoli Federico II; we acknowledge the support of the University.

References

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

[2] T. Ågotnes, V. Goranko, and W. Jamroga. Alternating-time temporal logics with irrevocable strategies. In *Proceedings of TARK XI*, pages 15–24, 2007.

[3] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge. Knowledge and ability. In H.P. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B.P. Kooi, editors, *Handbook of Epistemic Logic*, pages 543–589. College Publications, 2015.

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

[5] N. Alechina, N. Bulling, B. Logan, and H.N. Nguyen. The virtues of idleness: A decidable fragment of resource agent logic. *Artificial Intellegence*, 245:56–85, 2017.

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

[7] N. Alechina, B. Logan, M. Dastani, and J.-J. Ch. Meyer. Reasoning about agent execution strategies. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1455–1458, 2008.

[8] N. Alechina, B. Logan, N.H. Nga, and A. Rakib. A logic for coalitions with bounded resources. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 659–664, 2009.

[9] N. Alechina, B. Logan, H.N. Nguyen, and F. Raimondi. Model-checking for resource-bounded ATL with production and consumption of resources. *Journal of Computer and System Sciences*, 88:126–144, 2017.

[10] N. Alechina, B. Logan, H.N. Nguyen, and A. Rakib. Resource-bounded alternating-time temporal logic. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 481–488, 2010.

[11] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.

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

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

[14] E. Andre, W. Jamroga, M. Knapik, W. Penczek, and L. Petrucci. Timed atl: Forget memory, just count. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2017*, pages 1460–1462. IFAAMAS, 2017.

[15] M. Barlo, G. Carmona, and H. Sabourian. Bounded memory with finite action spaces. In *Third World Congress of the Game Theory Society (GAMES)*, 2008.

[16] F. Belardinelli, R. Condurache, C. Dima, W. Jamroga, and A.V. Jones. Bisimulations for verification of strategic abilities with application to ThreeBallot voting protocol. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2017*, pages 1286–1295. IFAAMAS, 2017.

[17] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multiagent systems with imperfect information and public actions. In *Proceedings of AAMAS*, pages 1268–1276, 2017.

[18] Francesco Belardinelli. Reasoning about knowledge and strategies: Epistemic strategy logic. In *Proceedings 2nd International Workshop on Strategic Reasoning, SR 2014, Grenoble, France, April 5-6, 2014.*, pages 27–33, 2014.

[19] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Alternating-time temporal logic on finite traces. In *IJCAI 2018*, pages 77–83, 2018.

[20] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of Computing*, pages 544–553. ACM, 1994.

[21] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Y. Vardi. Strategy logic with imperfect information. In *Proceedings of LICS*, pages 1–12, 2017.

[22] V. Bhaskar, G.J. Mailath, and S. Morris. A foundation for markov equilibria in sequential games with finite social memory. *Review of Economic Studies*, 80(20):925–948, 2012.

[23] A. Biere, A. Cimatti, E.M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proceedings of Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 1999.

[24] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, and B. Zanuttini. Boolean games revisited. In *Proceedings of ECAI*, pages 265–269, 2006.

[25] 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.

[26] R. H. Bordini, A. L. C. Bazzan, R. de Oliveira Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. Agentspeak(xl): efficient intention selection in BDI agents via decision-theoretic task scheduling. In *Proceedings of AAMAS*, pages 1294–1302, 2002.

[27] R. H. Bordini, M. Wooldridge, and J. F. Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons, 2007.

[28] L. E. Bourne. Knowing and using concepts. *Psychol. Rev.*, 77:546–556, 1970.

[29] R.I. Brafman, G. De Giacomo, and F. Patrizi. Specifying non-markovian rewards in mdps using LDL on finite traces (preliminary version). *CoRR*, abs/1706.08100, 2017.

[30] T. Brazdil, K. Chatterjee, M. Chmelik, A. Fellner, and J. Kretinsky. Counterexample explanation by learning small strategies in Markov Decision Processes. In *Proceedings of CAV*, pages 158–177, 2015.

[31] T. Brazdil, K. Chatterjee, J. Kretinsky, and V. Toman. Strategy representation by decision trees in reactive synthesis. In *Proceedings of TACAS*, 2018. To appear. Available at `http://pub.ist.ac.at/~vtoman/paper/TACAS_2018_paper_40.pdf`.

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

[33] T. Brihaye, A. Da Costa Lopes, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. In *Proceedings of LFCS*, volume 5407 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2009.

[34] N. Bulling and J. Dix. Modelling and verifying coalitions using argumentation and ATL. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 14(46):45–73, 2010.

[35] N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity. In M. Dastani, K. Hindriks, and J.-J. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.

[36] N. Bulling and B. Farwer. Expressing properties of resource-bounded systems: The logics RTL* and RTL. In *Proceedings of Computational Logic in Multi-Agent Systems (CLIMA)*, volume 6214 of *Lecture Notes in Computer Science*, pages 22–45, 2010.

[37] N. Bulling and B. Farwer. On the (un-)decidability of model checking resource-bounded agents. In *Proceedings of ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 567–572. IOS Press, 2010.

[38] N. Bulling and W. Jamroga. What agents can probably enforce. *Fundamenta Informaticae*, 93(1-3):81–96, 2009.

[39] N. Bulling and W. Jamroga. Verifying agents with memory is harder than it seemed. *AI Communications*, 23:380–403, 2010.

[40] N. Bulling and W. Jamroga. Comparing variants of strategic ability: How uncertainty and memory influence general properties of games. *Journal of Autonomous Agents and Multi-Agent Systems*, 28(3):474–518, 2014.

[41] N. Bulling, W. Jamroga, and J. Dix. Reasoning about temporal properties of rational play. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):51–114, 2008.

[42] 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.

[43] P. Čermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications. In *CAV'14*, LNCS 8559, pages 524–531. Springer, 2014.

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

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

[46] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A model checker for stochastic multi-player games. In *TACAS*, LNCS 7795, pages 185–191. Springer, 2013.

[47] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.

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

[49] E. Davis and G. Marcus. Commonsense reasoning. *Communications of the ACM*, 58(9):92–103, 2015.

[50] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *Theoretical Computer Science*, 345:139–170, 2005.

[51] L. de Alfaro and T.A. Henzinger. Concurrent omega-regular games. In *Proceedings of LICS*, pages 141–154, 2000.

[52] G. De Giacomo and M.Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of IJCAI*, pages 854–860, 2013.

[53] G. De Giacomo and M.Y. Vardi. Synthesis for LTL and LDL on finite traces. In *Proceedings of IJCAI*, pages 1558–1564, 2015.

[54] C. Dima and F.L. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[55] H. Duijf and J.M. Broersen. Representing strategies. In *Proceedings of SR*, pages 15–26, 2016.

[56] J. Feldman. Minimization of Boolean complexity in human concept learning. *Nature*, 407:630–3, 11 2000.

[57] W. Gelade and F. Neven. Succinctness of the complement and intersection of regular expressions. In *Proceedings of STACS*, pages 325–336, 2008.

[58] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[59] V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, 353(1):93–117, 2006.

[60] H. Gruber and M. Holzer. From finite automata to regular expressions and back - A summary on descriptional complexity. *International Journal of Foundations of Computer Science*, 26(8):1009–1040, 2015.

[61] A. Gupta, S. Schewe, and D. Wojtczak. Making the best of limited memory in multi-player discounted sum games. In *Proceedings of GandALF*, pages 16–30, 2015.

[62] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Iterated boolean games. *Information and Computation*, 242:53–79, 2015.

[63] J. Gutierrez, P. Harrenstein, and M. Wooldridge. From model checking to equilibrium checking: Reactive modules for rational verification. *Artificial Intelligence*, 248:123–157, 2017.

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

[65] P. Harrenstein. *Logic in Conflict*. PhD thesis, Utrecht University, 2004.

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

[67] K.V. Hindriks, F.S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[68] K.V. Hindriks, F.S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming with declarative goals. In *Proceedings of ATAL*, pages 228–243, 2000.

[69] John E. Hopcroft and Jeffrey D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley, 1990.

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

[71] X. Huang and R. van der Meyden. Symbolic model checking epistemic strategy logic. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 1426–1432, 2014.

[72] P. Jackson. *Introduction To Expert Systems (3 ed.)*. Addison Wesley, 1998.

[73] W. Jamroga. A temporal logic for stochastic multi-agent systems. In *Proceedings of PRIMA'08*, volume 5357 of *Lecture Notes in Computer Science*, pages 239–250, 2008.

[74] W. Jamroga and T. Ågotnes. Constructive knowledge: What agents can achieve under incomplete information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, 2007.

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

[76] W. Jamroga and J. Dix. Model checking abilities of agents: A closer look. *Theory of Computing Systems*, 42(3):366–410, 2008.

[77] W. Jamroga, M. Knapik, and D. Kurpiewski. Fixpoint approximation of strategic abilities under imperfect information. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1241–1249. IFAAMAS, 2017.

[78] W. Jamroga, V. Malvone, and A. Murano. Reasoning about natural strategic ability. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 714–722. IFAAMAS, 2017.

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

[80] M. Kacprzak, A. Lomuscio, and W. Penczek. From bounded to unbounded model checking for temporal epistemic logic. volume 63, pages 221–240, 2004.

[81] N.R. Kocherlakota. Money is memory. *Journal of Economic Theory*, 81(2):232–251, 1998.

[82] O. Kupferman, N. Piterman, and M.Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.

[83] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: probabilistic symbolic model checker. In *Proceedings of TOOLS*, volume 2324 of *Lecture Notes in Computer Science*, pages 200–204. Springer, 2002.

[84] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *Logical Methods in Computer Science*, 4(7), 2008.

[85] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 2015. Availabe online.

[86] A. Lomuscio and F. Raimondi. MCMAS : A model checker for multi-agent systems. In *TACAS*, LNCS 4314, pages 450–454. Springer, 2006.

[87] V. Malvone, A. Murano, and L. Sorrentino. Hiding actions in multi-player games. In *Proceedings of AAMAS*, pages 1205–1213, 2017.

[88] N. Markey and P. Schnoebelen. Model checking a path. In *International Conference on Concurrency Theory*, pages 251–265. Springer, 2003.

[89] F. Mogavero, A. Murano, G. Perelli, , and M.Y. Vardi. What makes ATL* decidable? a decidable fragment of strategy logic. In *Proceedings of CONCUR*, pages 193–208, 2012.

[90] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic*, 15(4):1–42, 2014.

[91] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: on the satisfiability problem. *Logical Methods in Computer Science*, 13(1), 2017.

[92] F. Mogavero, A. Murano, and L. Sorrentino. On promptness in parity games. *Fundamenta Informaticae*, 139(3):277–305, 2015.

[93] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning about strategies. In *Proceedings of FSTTCS*, pages 133–144, 2010.

[94] R. Morris and G. Ward. *The Cognitive Psychology of Planning*. Psychology Press, 2014.

[95] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

[96] N.J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158, 1994.

[97] P. Novák and W. Jamroga. Code patterns for agent oriented programming. In *Proceedings of AAMAS'09*, pages 105–112, 2009.

[98] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[99] S. Paul and R. Ramanujam. Imitation in large games. In *Proceedings of Games, Automata, Logics and Formal Verification (GandALF)*, pages 162–172, 2010.

[100] S. Paul, R. Ramanujam, and S.E. Simon. Stability under strategy switching. In *Mathematical Theory and Computational Practice, Proceedings of CiE*, pages 389–398, 2009.

[101] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 209–216, New York, NY, USA, 2003. ACM Press.

[102] R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press : Cambridge, Mass, 1999.

[103] J.P. Queille and J. Sifakis. Specification and verification of concurrent programs in Cesar. In *Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 1981.

[104] M.O. Rabin and D.S. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[105] A. Rubinstein. *Modeling bounded rationality*. MIT Press, 1998.

[106] S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice Hall, 1995.

[107] F.P. Santos. *Dynamics of Reputation and the Self-organization of Cooperation*. PhD thesis, University of Lisbon, 2018.

[108] F.P. Santos, F.C. Santos, and J.M. Pacheco. Social norm complexity and past reputations in the evolution of cooperation. *Nature*, 555:242–245, 2018.

[109] Sven Schewe. ATL* satisfiability is 2ExpTime-complete. In *Proceedings of ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 373–385. Springer, 2008.

[110] Ph. Schnoebelen. The complexity of temporal model checking. In *Advances in Modal Logics, Proceedings of AiML 2002*. World Scientific, 2003.

[111] H. Schnoor. Deciding epistemic and strategic properties of cryptographic protocols. In *Proceedings of ESORICS*, pages 91–108, 2012.

[112] Henning Schnoor. Strategic planning for probabilistic games with incomplete information. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1057–1064, 2010.

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

[114] Y. Shoham and K. Leyton-Brown. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[115] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[116] A. Toumi, J. Gutierrez, and M. Wooldridge. A tool for the automated verification of nash equilibria in concurrent games. In *Proceedings of ICTAC*, volume 9399 of *Lecture Notes in Computer Science*, pages 583–594. Springer, 2015.

[117] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *Proceedings of AAMAS'05*, pages 157–164, 2005.

[118] W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 201–208. ACM, 2006.

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

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

[121] S. van Otterloo and G. Jonker. On Epistemic Temporal Strategic Logic. *Electronic Notes in Theoretical Computer Science*, 126:77–92, 2004. Proceedings of LCMAS'04.

[122] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Preferences in game logics. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 152–159, 2004.

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

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

[125] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press: Princeton, NJ, 1944.

[126] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. ATL satisfiability is indeed EXPTIME-complete. *Journal of Logic and Computation*, 16(6):765–787, 2006.

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

[128] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational verification: From model checking to equilibrium checking. In *Proceedings of AAAI*, pages 4184–4191, 2016.

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

Appendix A. Proof of Theorem 9

We start by proving the upper bound.

Proposition 35. *Model checking* NatATL$_r$ *is in* $\mathbf{\Delta_2^P}$ *with respect to the size of the model, the length of the formula, and the maximal bound* k *in the formula.*

Proof. We make use of a bottom-up procedure based on the one introduced in the proof of Theorem 5. Precisely, take an arbitrary formula φ of NatATL$_r$ 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 Proposition 6 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$_r$ is $\mathbf{P^{NP}} = \mathbf{\Delta_2^P}$. □

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

Definition 16. *Given a fixed number* r *and* $1 \leq i \leq r$*, an 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 *in CNF involving only on variables in* $X_i \cup \{z_1, \ldots, z_{i-1}\}$*;*

Variable z_i *is assigned* $true$ *if there exists an assignment of variables in* X_i *such that* φ_i *is true; otherwise* z_i *is assigned* $false$*. By a slight abuse of notation, we can write it as* $z_i \equiv \exists X_i \varphi_i(z_1, \ldots, z_{i-1}, X_i)$*. The output of the SNSAT instance is the truth value of* z_r*.*

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, $M_i = (\mathbb{A}\mathrm{gt}, St^i, Act^i, d^i, t^i, Prop^i, V^i)$, where:

- $\mathbb{A}\mathrm{gt} = \{\mathbf{v}, \mathbf{r}\}$,
- $St^i = \{q_0^i\} \cup St_{cl} \cup St_{prop} \cup St_{neg} \cup \{q_\top\} \cup St^{i-1}$, 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 $St_{neg} = \{neg_1^i, \ldots, neg_{i-1}^i\}$;
- $Act^i = \{I, C_1, \ldots, C_n, x_1, \ldots, x_m, z_1, \ldots, z_r, \top, \bot\}$;
- $d^i(\mathbf{v}, q_0^i) = d^i(\mathbf{v}, q_\top) = d^i(\mathbf{v}, neg_h^i) = \{I\}$, $d^i(\mathbf{v}, q_j^i) = \{x_k \mid x_{k,i}$ or $\neg x_{k,i}$ is in $C_j^i\} \cup \{z_h \mid z_h$ or $\neg z_h$ is in $C_j^i\}$, $d^i(\mathbf{v}, q_{j,k}^i) = \{\top, \bot\}$; $d^i(\mathbf{r}, q_0^i) = \{C_1, \ldots, C_n\}$ 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}: $t^i(q, \alpha_1, \alpha_2) = t^{i-1}(q, \alpha_1, \alpha_2)$;
- $Prop^i = \{\mathsf{C}_1^i, \ldots, \mathsf{C}_n^i, \mathsf{x}_1^i, \ldots, \mathsf{x}_m^i, \mathsf{win}, \mathsf{neg}\}$;
- $V(q_0^i) = \emptyset$, $V(q_j^i) = \mathsf{C}_\mathsf{j}^\mathsf{i}$, $V(q_{j,k}^i) = \mathsf{x}_\mathsf{k}^\mathsf{i}$, $V(neg_h^i) = \mathsf{neg}$ and $V(q_\top) = \mathsf{win}$.

where $1 \leq i \leq r$, $1 \leq j \leq n$, $1 \leq k \leq m$, and $1 \leq h < i$. An example model is presented in Figure A.18.

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

$$\Psi_1 = \langle\langle \mathbf{v} \rangle\rangle^{\leq n+m}(\neg\mathsf{neg})\,\mathsf{U}\,(\mathsf{win}),$$
$$\ldots$$
$$\Psi_i = \langle\langle \mathbf{v} \rangle\rangle^{\leq n+m}(\neg\mathsf{neg})\,\mathsf{U}\,(\mathsf{win} \vee (\mathsf{neg} \wedge \langle\langle \emptyset \rangle\rangle^{\leq 0}\mathsf{X}\,\neg\Psi_{i-1})).^{10}$$

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

Lemma 36. *$\forall 1 \leq l \leq i \leq r$: $M_l, q_0^l \models \Psi_i$ iff $M_l, q_0^l \models \Psi_{i-1}$.*

The proof of the lemma is a straightforward adaptation of [75, Lemma 5].

Proposition 37. *$\forall 1 \leq i \leq r$: z_i is true iff $M_i, q_0^i \models \Psi_i$*

[10] Recall that $\langle\langle \emptyset \rangle\rangle^{\leq 0}$ is equivalent to the CTL path quantifier A ("for all paths"), cf. Proposition 2.

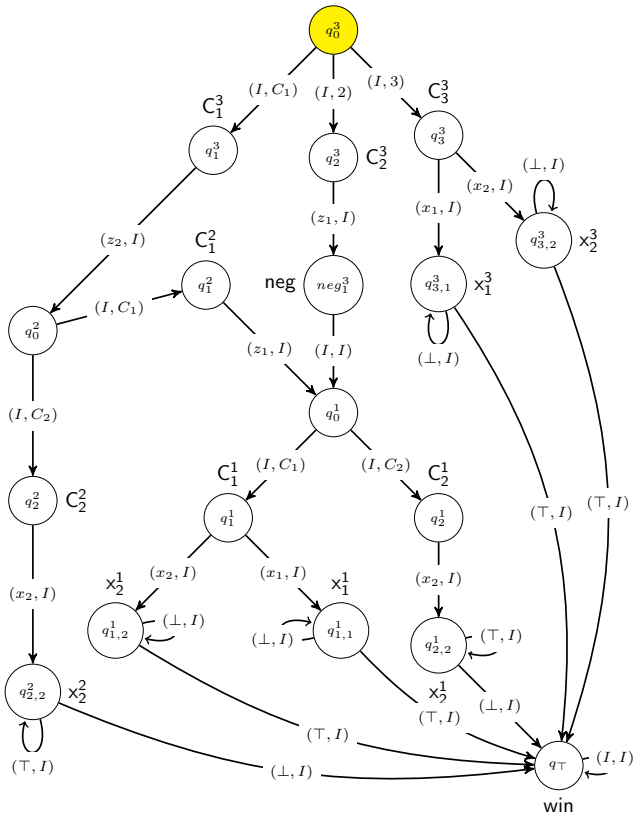

Figure A.18: CGS M_3 for $\varphi_3 = z_2 \wedge \neg z_1 \wedge (x_1 \vee x_2)$, $\varphi_2 = z_1 \wedge \neg x_2$, and $\varphi_1 = (x_1 \vee x_2) \wedge \neg x_2$. For simplicity, we omit the states that have no ingoing edges

Proof. Induction on i:

(i) For $i = 1$, we use the proof of Proposition 7.

(ii) For $i > 1$, we prove both directions.

(⇒) Firstly, if z_i is true then there is a valuation υ of X_i that makes φ_i true. We construct $s_{\mathbf{v}}$ as in the proof of Proposition 7. In case that some $x^s_{k,i}$ has been chosen in clause C^i_j then we define the guarded action $(\mathsf{C}^{\mathsf{i}}_{\mathsf{j}},\ x_k)$ and we are done. In case that some z^-_h has been chosen in clause C^i_j, where $h < i$, we have (by induction) that $M_h, q^h_0 \models \neg\Psi_h$. By Lemma 36, also $M_h, q^h_0 \models \neg\Psi_i$, and hence $M_i, q^h_0 \models \neg\Psi_i$. So we can make the same choice (i.e., we define the guarded action $(\mathsf{C}^{\mathsf{i}}_{\mathsf{j}},\ z_h)$) in $s_{\mathbf{v}}$, and this will lead to state neq^i_h, in which it holds that $neg \wedge A\mathsf{X}\,\neg\Psi_i$. In case that some z^+_h has been chosen in clause C^i_j, we have that $M_h, q^h_0 \models \Psi_h$. By Lemma 36, also $M_h, q^h_0 \models \Psi_i$. That is, there is a strategy $s'_{\mathbf{v}}$ in M_h such that $(\neg\mathsf{neg})\ \mathsf{U}\ (\mathsf{win} \vee (\mathsf{neg} \wedge A\mathsf{X}\,\neg\Psi_{i-1}))$ holds for all paths from $out(q^h_0, s'_{\mathbf{v}})$. Then, we can merge $s'_{\mathbf{v}}$ into $s_{\mathbf{v}}$.

(⇐) Conversely, if $M_i, q^i_0 \models \Psi_i$, then there is a strategy $s_{\mathbf{v}}$ that enforces $(\neg\mathsf{neg})\ \mathsf{U}\ ($

Algorithm $ExistNash(G, k)$:

1 $s_{\mathbb{A}\mathrm{gt}} = GuessStrat(G, \mathbb{A}\mathrm{gt}, k)$;
2 **return** (**not** $IsNotNash(G, s_{\mathbb{A}\mathrm{gt}}, k)$);

Figure A.19: Algorithm to decide EXISTNASH.

$\mathsf{win} \vee (\mathsf{neg} \wedge A\mathsf{X}\,\neg\Psi_{i-1})$). First, we consider the clause C^i_j with guarded action $(\mathsf{C}^{\mathsf{i}}_{\mathsf{j}},\ x_k)$, i.e. for which a propositional state is chosen by $s_{\mathbf{v}}$. The strategy defines a valuation for X_i that satisfies these clauses. For the other clauses, i.e. there is a guarded action $(\mathsf{C}^{\mathsf{i}}_{\mathsf{j}},\ z_h)$, we have two possibilities:

- $s_{\mathbf{v}}$ chooses q^h_0 in the state corresponding to C^i_j. Neither win nor neg have been encountered on this path yet, so we can take $s_{\mathbf{v}}$ to demonstrate that $M_i, q^h_0 \models \Psi_i$, and hence $M_h, q^h_0 \models \Psi_i$. By Lemma 36, also $M_h, q^h_0 \models \Psi_h$. By induction, z_h must be true, and hence clause C^i_j is satisfied.
- $s_{\mathbf{v}}$ chooses neg^i_h in the state corresponding to C^i_j. Then, it must be that $M_i, neg^i_h \models A\mathsf{X}\,\neg\Psi_{i-1}$, and hence $M_h, q^h_0 \models \neg\Psi_{i-1}$. By Lemma 36, also $M_h, q^h_0 \models \neg\Psi_h$. By induction, z_h must be false, and hence clause C^i_j (containing $\neg z_h$) is also satisfied.

□

By Propositions 35 and 37, we get the following result.

Theorem 9. *Model checking* NatATL$_r$ *is* $\mathbf{\Delta_2^P}$*-complete with respect to the size of the model, the length of the formula, and the maximal bound* k *in the formula.*

Appendix B. Proof of Theorem 29

Proposition 38. EXISTNASH *is in* $\mathbf{\Sigma_2^P}$.

Proof. A nondeterministic polynomial-time algorithm to decide EXISTNASH is presented in Figure A.19. The algorithm calls the procedure $IsNotNash$ in Figure 12 as an oracle. □

Proposition 39. EXISTNASH *is* $\mathbf{\Sigma_2^P}$*-hard.*

Proof. To prove the hardness, we extend the reduction used in the proof of Proposition 27. The idea is to add a new agent $\mathbf{d}$ (the "decider") whose sole role is to make sure that no strategy profile where $\mathbf{v}$ loses can be a Nash equilibrium. Then the only Nash equilibria that are left must involve a surely winning strategy for the verifier. This is achieved by enhancing the CGS by two additional segments, where the refuter and the decider play "matching pennies" if $tr(\varphi)$ does not hold. That is, first the refuter chooses "heads" (w_1) or "tails" ($\neg\mathsf{w}_1$); then, the decider does the same, choosing w_2

or $\neg \mathsf{w}_2$. The objectives are set as follows: $\Phi_{\mathbf{v}} = tr(\varphi)$, i.e., $\mathbf{v}$ wants to get ψ satisfied; $\Phi_{\mathbf{r}} = \neg tr(\varphi) \wedge (\mathrm{F}w_1 \leftrightarrow \mathrm{F}w_2)$, i.e., $\mathbf{r}$ wants to invalidate ψ and to match the "pennies;" $\Phi_{\mathbf{d}} = \neg(\mathrm{F}\mathsf{w}_1 \leftrightarrow \mathrm{F}\mathsf{w}_2)$, i.e., $\mathbf{d}$ wants to get a mismatch. Clearly, when $\neg tr(\varphi)$ holds, there is no equilibrium, since for every strategy profile that gives a match (and win of the refuter), the decider is better off changing his strategy, and vice versa. Thus, equilibrium can only exist when $tr(\varphi)$ holds, and hence $\Phi_{\mathbf{v}} = true$ and $\Phi_{\mathbf{r}} = false$. Conversely, $\Phi_{\mathbf{v}} = true$, $\Phi_{\mathbf{r}} = false$, and $\Phi_{\mathbf{d}} = true$ is sufficient for the strategy profile to be in equilibrium. Analogously to the proof of Proposition 27, we conclude that $\mathrm{QBF}_2(\psi, n, m) = true$ iff $\textsc{ExistNash}(G, max(n, m-n)) = true$.

Formally, let $\varphi = \exists x_1 \dots x_n \forall x_{n+1} \dots x_m \psi$ be a QBF_2 formula in negation normal form. We construct $G = (M, q_0, \Phi)$ with $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V \rangle$, where:

- $\mathbb{A}\mathrm{gt} = \{\mathbf{v}, \mathbf{r}, \mathbf{d}\}$;
- $St = \{x_1, \dots, x_m\} \cup \{p_1, \dots, p_m\} \cup \{\bar{p}_1, \dots, \bar{p}_m\} \cup \{w_1, w_2\} \cup \{\bar{w}_1, \bar{w}_2\} \cup \{chk_{\mathbf{r}}, chk_{\mathbf{d}}\} \cup \{end\}$;
- $Act = \{\top, \bot, I\}$;
- $d_{\mathbf{v}}(s_i) = \begin{cases} \{\top, \bot\} & \text{if } s_i = x_i \text{ and } 1 \leq i \leq n; \\ \{I\} & \text{otherwise.} \end{cases}$

  $d_{\mathbf{r}}(s_i) = \begin{cases} \{\top, \bot\} & \text{if } s_i = x_i \text{ or } s_i = chk_{\mathbf{r}} \text{ and } n+1 \leq i \leq m; \\ \{I\} & \text{otherwise.} \end{cases}$

  $d_{\mathbf{d}}(s_i) = \begin{cases} \{\top, \bot\} & \text{if } s_i = chk_{\mathbf{d}}; \\ \{I\} & \text{otherwise.} \end{cases}$
- $t(x_i, (\top, I, I)) = p_i$, $t(x_i, (\bot, I, I)) = \bar{p}_i$, $t(x_j, (I, \top, I)) = p_j$, $t(x_j, (I, \bot, I)) = \bar{p}_j$, $t(s_k, (I, I, I)) = x_{k+1}$, $t(chk_{\mathbf{r}}, (I, \top, I)) = w_1$, $t(chk_{\mathbf{r}}, (I, \bot, I)) = \bar{w}_1$, $t(chk_{\mathbf{d}}, (I, I, \top)) = w_2$, $t(chk_{\mathbf{d}}, (I, I, \bot)) = \bar{w}_2$, and $t(s, (I, I, I)) = end$, where $s_k \in \{p_k, \bar{p}_k\}$, $s \in \{w_2, \bar{w}_2, end\}$, $1 \leq i \leq n$, $n+1 \leq j \leq m$, and $1 \leq k \leq m$.
- $Prop = \{\mathsf{x}_1, \dots, \mathsf{x}_m\} \cup \{\mathsf{p}_1, \dots, \mathsf{p}_m\} \cup \{\bar{\mathsf{p}}_1, \dots, \bar{\mathsf{p}}_m, \mathsf{w}_1, \mathsf{w}_2\}$;
- $V(x_i) = \{\mathsf{x}_i\}$, $V(p_i) = \{\mathsf{p}_i\}$, $V(\bar{p}_i) = \{\bar{\mathsf{p}}_i\}$, $V(w_i) = \{\mathsf{w}_i\}$, and $V(chk_{\mathbf{r}}) = V(chk_{\mathbf{d}}) = V(\bar{w}_i) = V(end) = \emptyset$;
- $q_0 = x_1$;
- $\Phi = \{\Phi_{\mathbf{v}}, \Phi_{\mathbf{r}}, \Phi_{\mathbf{d}}\}$, such that $\Phi_{\mathbf{v}} = tr(\varphi)$, $\Phi_{\mathbf{r}} = \neg tr(\varphi) \wedge (\mathrm{F}w_1 \leftrightarrow \mathrm{F}w_2)$, and $\Phi_{\mathbf{d}} = \neg(\mathrm{F}\mathsf{w}_1 \leftrightarrow \mathrm{F}\mathsf{w}_2)$, where $tr(\varphi) = \psi[\mathrm{F}\bar{\mathsf{p}}_i/\neg x_i, \mathrm{F}\mathsf{p}_i/x_i]$.

Figure B.20 presents the game G for the QBF_2 formula $\varphi = \exists x_1 x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$.

To conclude the proof, it remains to show that the construction is correct, that is $\mathrm{QBF}_2(\psi, n, m) = true$ iff $\textsc{ExistNash}(G, max(n, m-n)) = true$. For this, one can use a reasoning similar to the one in the proof of Proposition 27. □

By putting together Propositions 38 and 39, we get the following.

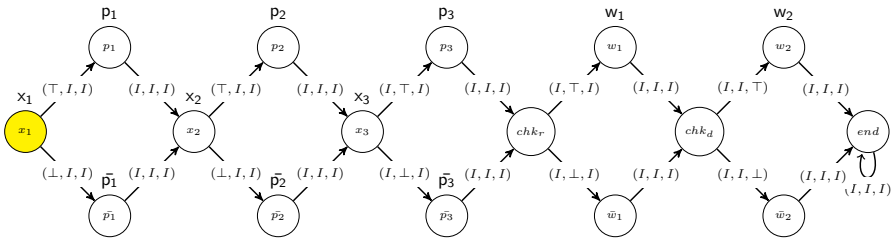

Figure B.20: The concurrent game structure corresponding to the QBF_2 formula $\varphi = \exists x_1 x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$ obtained by the construction of Proposition 39. The objectives are: $\Phi_{\mathbf{v}} = (\mathrm{F}\mathsf{p}_1 \wedge \mathrm{F}\bar{\mathsf{p}}_2) \vee \mathrm{F}\mathsf{p}_3$, $\Phi_{\mathbf{r}} = \neg((\mathrm{F}\mathsf{p}_1 \wedge \mathrm{F}\bar{\mathsf{p}}_2) \vee \mathrm{F}\mathsf{p}_3) \wedge (\mathrm{F}w_1 \leftrightarrow \mathrm{F}w_2)$, $\Phi_{\mathbf{d}} = \neg(\mathrm{F}w_1 \leftrightarrow \mathrm{F}w_2)$.

Theorem 29. EXISTNASH *is* $\boldsymbol{\Sigma_2^{\mathrm{P}}}$*-complete with respect to the size of the game and the value of the bound* k*.*