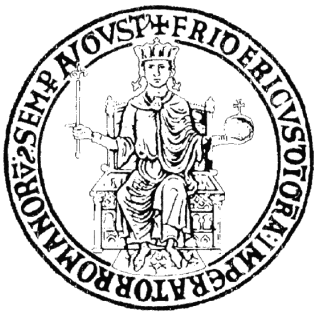
Strategic Reasoning in Game Theory



Vadim Malvone

Università degli studi di Napoli "Federico II"
Dipartimento di Matematica e Applicazioni "R. Caccioppoli"

Dottorato in **Scienze Matematiche e Informatiche**
Ciclo XXX

A thesis submitted in fulfillment of the degree of
Doctor in Compute Science

Submission: December 11, 2017
Defense: Napoli, February -, 2018
Revised version: December 11, 2017

Supervisor: Prof. Aniello Murano

Contents

Abstract

Game theory in AI is a powerful mathematical framework largely applied in the last three decades for the strategic reasoning in multi-agent systems. Seminal works along this line started with turn-based two-player games (under perfect and imperfect information) to check the correctness of a system against an unpredictable environment. Then, large effort has been devoted to extend those works to the multi-agent setting and, specifically, to efficiently reasoning about important solution concepts such as Nash Equilibria and the like. Breakthrough contributions along this direction concern the introduction of logics for the strategic reasoning such as Alternating-time Temporal Logic (ATL), Strategy Logic (SL), and their extensions.

Two-player games and logics for the strategic reasoning are nowadays very active areas of research. In this thesis we significantly advance the work along both these two lines of research by providing fresh studies and results of practical application.

We start with two-player reachability games and first investigate the problem of checking whether a designed player has more than a winning strategy to reach a target. We investigate this question under both perfect and imperfect information. We provide an automata-based solution that requires linear-time, in the perfect information setting, and exponential-time, in the imperfect one. In both cases, the results are tight. Then, we move to multi-player concurrent games and study the following specific setting: (i) $Player_0$'s objective is to reach a target W, and (ii) the opponents are trying to stop this but have partial observation about $Player_0$'s actions. We study the problem of deciding whether the opponents can prevent $Player_0$ to reach W. We show, using an automata-theoretic approach that, assuming the opponents have the same partial observation and play under uniformity, the problem is in EXPTIME. We recall that, in general, multi-player reachability games with imperfect information are undecidable.

Then, we move to the more expressive framework of logics for the strategic reasoning. We first introduce and study two graded extensions of SL, namely GSL and GRADEDSL. By the former, we define a graded version over single strategy variables, i.e. "there exist at least g different strategies", where the strategies are counted semantically. We study the model checking-problem for GSL and show that for its fragment called vanilla GSL[1G] the problem is PTIME-complete. By GRADEDSL, we consider a graded version over tuple of strategy variables and use a syntactic counting over strategies. By means of GRADEDSL we show how to count the number of different strategy profiles that are Nash equilibria (NE). By analyzing the structure of the specific formulas involved, we conclude that the important problem of checking for the existence of a unique NE can be solved in 2EXPTIME, which is not harder than merely checking for the existence of such an equilibrium. Finally, we adopt the view of bounded rationality, and look only at "simple" strategies in specifications of

agents' abilities. We formally define what "simple" means, and propose a variant of plain ATL, namely NatATL, that takes only such strategies into account. We study the model checking problem for the resulting semantics of ability and obtain tight results. The positive results achieved with NatATL encourage for the investigation of simple strategies over more powerful logics, including SL.

Introduction

Game theory [WJ95] in AI is a powerful mathematical framework to reason about *reactive* systems [HP85]. These systems are characterized by an ongoing interaction between two or more entities, modeled as *players*, and the behavior of the entire system deeply relies on this interaction [HP85]. Game theory has been largely investigated in a number of different fields. In economics, it is used to deal with *solution concepts* such as Nash Equilibrium [Mye91]. In biology, it is used to reason about the *phenotypic evolution* [Smi82]. In computer science, it is applied to solve problems in robotics, multi-agent system verification and planning [KVW01, JM14, Woo02].

In the basic setting, a (finite) game consists of two players, conventionally named $Player_0$ and $Player_1$, playing a finite number of times, in a turn-based manner, i.e., the moves of the players are interleaved. Technically, the configurations (states) of the game are partitioned between $Player_0$ and $Player_1$ and a player moves in a state whenever he owns it. Solving a two-player game amounts to check whether $Player_0$ has a *winning strategy*. That is, to check whether he can take a sequence of move *actions* (a *strategy*) that allows him to satisfy the game objective, no matter how his opponent plays.

Depending on the visibility the players have over the game, we distinguish between *perfect* and *imperfect* information games [Rei84]. The former corresponds to a very basic setting in which every player has full knowledge about the game arena and the moves taken by the other players. However, such a game model has rare applications in real-life scenarios where it is common to have situations in which a player has to come to a decision without having all relevant information at hand. In computer science this situation occurs for example when some variables of a system are internal/private and not visible to an external environment [KV97, BCJK15]. For instance, consider an ATM and a customer player aiming to withdraw some money. At a certain point, the controller player internally decides the denominations of the bills to delivery to the customer player and this is revealed only at the end of the interaction between the two actors, that is when the game ends.

In game models for AI, the imperfect information is usually modeled by setting an indistinguishability relation over the states of the arena [KV97, Rei84, PR90]. In this case, during a play, it may happen that some players cannot tell precisely in which state they are, but rather they observe a set of states. Therefore these players cannot base their strategy on the exact current situation. This means that over indistinguishable sets they can only use the same move or, similarly, that some perfect information moves are not valid anymore. This constraint deeply impacts on the problem of deciding who wins the game. Indeed, it is well known that multi-player games of imperfect information are computationally hard and, in some cases they become non-elementary or even undecidable [PR89, Rei84].

An important application of game theory in computer science and, more recently, in AI, concerns formal-system verification [CE81, CGP02, KVW00, QS82]. In particular, game theory has come to the fore as a powerful tool for the verification of reactive systems and embedded systems. This story of success goes back to late seventies with the introduction of the *model checking* technique [CE81, QS82] by Clarke, Emerson, Sifakis, and Quielle for which the first three authors received the prestigious Turing award prize. The idea of model checking is powerful and simple at the same time: to check whether a system satisfies a desired behavior we check instead, by means of a suitable algorithm, whether a mathematical model of the system *meets* a formal specification describing the systems [CGP02]. For the latter, we usually use temporal logics such as LTL [Pnu77], CTL [CE81], CTL* [EH86], and the like. In particular LTL was introduced by Pnueli, who also got for this the Turing award.

Notably, first applications of model checking just concerned *closed systems*, which are characterized by the fact that their behavior is completely determined by their internal states. This makes the modeling quite easy, indeed one can simple use *Kripke structures*, that are *labeled-state transition systems*, and the verification process is also easy as we have to handle with only one source of non-determinism, *i.e.*, the one coming from the system itself. Overall, it turns out that the model checking problem for closed systems with respect to LTL and CTL* specifications is PSPACE-complete, while it is just PTIME-complete for CTL specifications.

Unfortunately, all model checking techniques developed to handle closed systems turn out to be quite useless in practice as most of the systems are *open* in the sense that they are characterized by an ongoing interaction with an external environment on which the whole behavior of the system relies. This makes the verification process much harder as one has to deal with two sources of non-determinism, one coming from the environment and one coming from the system itself. Also, to model open systems we need more involved structures (than Kripke structures) in which one has to explicitly take into consideration the interaction between the system and the external environment. To overcome this problem, Kupferman, Vardi and Wolper introduced in late nineties *module checking* [KVW01] a specific framework to handle the verification of open systems against branching-time temporal-logics such as CTL, CTL* and the like. In particular, they showed that the verification procedure for open systems always requires an additional (and unavoidable) exponential-time complexity with respect to closed systems. Since its introduction, module checking has been a very active field of research and applied in several directions. Among the others we recall applications in the infinite-state recursive setting (i.e., pushdown systems) [BMP10, FMP08], as well as hierarchical systems [MNP08]. Module checking has been also investigated in the imperfect information setting [KV97, ALM$^+$13]. In particular, for finite-state systems the problem remains decidable although it requires an additional exponential-time complexity with respect to the size of the system [KV97].

Following the success of module checking, researchers started looking at more general open settings and, in particular, to *multi-agent systems* [Woo02, AHK02, JvdH04, CHP10, MMV10a, MMPV14]. These are systems whose behavior depends on the ongoing interaction between several autonomous entities (namely, *agents* or *players*) continuously interacting among them in a cooperative or adversarial manner, trying to achieve a designed goal. One of the most important developments in this field comes from Alur, Kupferman, and Henzinger, who introduced the logic ATL and its extension ATL* [AHK02]. ATL* allows to reason about strategies of agents having the satisfaction of temporal goals as payoff criterion. Formally, this logic is obtained as a generalization of CTL*, in which the existential $\mathtt{E}$ and the universal $\mathtt{A}$ *path quantifiers* are replaced with *strategic modalities* of the form $\langle\langle \mathrm{A} \rangle\rangle$ and $[\![\mathrm{A}]\!]$, where A is a set of *agents*. As far as the game model regards, *concurrent game structures* (in short CGS) are generally used. CGS are labeled-state transition graphs whose transitions are labeled with agent's decisions, *i.e.*, a tuple of actions, one for each agent. A player's strategy in a CGS can be seen as a "conditional plan" for possible moves. Formally it is defined as a function from a sequence of system states (i.e., possible histories of the game) to actions. This general definition of strategy is called *memoryfull* or *perfect recall* as a player can look at the entire past history of a play in order to come to a decision. Conversely, one can also consider a simpler notion of *positional* a.k.a. *memoryless* strategy, which is defined as a function from states to actions. Positional strategies are much easier to handle along the verification process. Indeed ATL* model checking is 2EXPTIME-complete for memoryfull strategies and PSPACE-complete for memoryless strategies (for ATL the complexity does not change and it is PTIME-complete). The difference becomes even more evident in the imperfect information setting. Indeed the model checking problem is undecidable already in the restricted case of ATL specifications with three players [DT11]. This is undesirable since imperfect information and memoryfull strategies (coming together) are quite common in real-life multi-agent settings. This has therefore limited the application of ATL and ATL* in practice. This problem has been recovered only recently by considering specific but realistic restrictions on the architecture of the game model [BMM17] or on the way the agents can hide their information along a play [BLMR17]. Under this restriction it has been possible to retain decidability. ATL (and its extension ATL*) has been largely studied in the last two decades, both from a practical and a theoretical point of view. For several years it has been considered the referred logic for the strategic reasoning in multi-agent systems. Tools based on ATL and ATL* also exist and the most famous one is MCMAS [LR06a, LQR09].

Despite its expressiveness, ATL* suffers from the strong limitation that strategies are treated only implicitly in the semantics of its modalities. This restriction makes the logic less suited to formalize important solution concepts, such as the *Nash Equilibrium*. These considerations led to the introduction and study of *Strategy Logic* (SL, for short) [CHP07, MMV10a], a more powerful formalism for the strategic reasoning. As a key aspect, this logic

treats strategies as *first-order objects* that can be determined by means of the existential $\langle\langle x \rangle\rangle$ and universal $[\![x]\!]$ quantifiers, which can be respectively read as *"there exists a strategy x"* and *"for all strategies x"*. Remarkably, in SL [MMV10a], a strategy is a generic conditional plan that at each step prescribes an action on the base of the history of the play. Therefore, this plan is not intrinsically glued to a specific agent, but an explicit binding operator (a, x) allows to link an agent a to the strategy associated with a variable x. Unfortunately, the high expressivity of SL comes at a price. Indeed, it has been shown that the model-checking problem for SL becomes non-elementary complete. To gain back elementariness, several fragments of SL, strictly subsuming ATL* have been considered. Among the others, One-Goal Strategy Logic (SG[1G], for short) considers SL formulas in a special prenex normal form having a single temporal goal at a time. For a goal, it is specifically meant a sequence of bindings followed by a temporal logic formula. It has been shown that for SG[1G] the model checking question is 2-EXPTIME-COMPLETE, as it is for ATL. If one allows for a Boolean combination of goals, then the resulting logic is named Boolean goal Strategy Logic (SG[BG]), for which the model checking problem is non-elementary-complete [BGM15]. SL is most recent and promising logic to be used for future applications in the strategic reasoning for multi-agent systems. Tools for this logic also exist [ČLMM14, ČLM15].

For what we have discussed so far, it is clear that two ingredients play a central role in formal verification multi-agent systems: the nature of the strategies and how much imperfect information among players it is admitted. In this thesis we carefully investigate both this issues under specific settings. As far as the strategies concerns, observe that in all cases discussed above, in order to win a game, we just look for the existence of a winning strategy for some coalitions of players or for its complement. This corresponds and extends the classical modalities $\exists$ and $\forall$ in temporal logic. Sometimes, however, it is convenient to have a more quantitative information about the number of strategies that allow to satisfy/not satisfy a given goal. For example, in Nash Equilibrium, such an information amounts to solve the challenging question of checking whether the equilibrium is unique [AKH02, PC79, CHS99, ORS93, BBV86, Fra92, GK93, MMMS15, AMMR16]. This problem impacts on the predictive power of Nash Equilibrium since, in case there are multiple equilibria, the outcome of the game cannot be uniquely pinned down [SLCB13, ZG11, Pav12]. As another example, consider the setting of robot rescue planning [KTN$^+$99, Kit00, KT01, CTC07]. It is not hard to imagine situations in which it is vital to know in advance whether a robot team has more than a winning strategy from a critical stage, just to have a backup plan in case an execution of a planned winning strategy cannot be executed anymore. Such a redundancy allows to strengthen the ability of winning the game and, specifically, the rescue capability.

Another aspect to consider about strategies is how much memory they have. In particular, memoryless strategies are too poor to cover real situations while memoryfull strategies make sense only from a mathematical point of view and in case we think of strategic ability of

a machine (robot, computer program). However, the latter kind of strategies are not very realistic for reasoning about human behavior. This is because humans are very bad at handling combinatorially complex objects. A human strategy should be relatively simple and "intuitive" or "natural" in order for the person to understand it, memorize it, and execute it. This applies even more if the human agent has to come up with the strategy on its own.

Regarding the imperfect information aspects, as far as we have discussed, it easily complicates the decision problem leading to non-elementariness or even to undecidability. In this thesis we have considered restricted but realistic multi-agent scenarios in which the verification problem of reaching a specific target is just EXPTIME-complete.

Now, we have all ingredients to show the problems and the related results of this thesis.

In Chapter 1 [1], we study the problem of checking whether, in a two-player reachability game, a designed player has more than a winning strategy. We investigate this question both under perfect and imperfect information about the moves performed by the players. We provide an automata-based solution that results, in the perfect information setting, in a linear-time procedure; in the imperfect information setting, instead, it shows an exponential-time upper bound. In both cases, the results are tight.

In Chapter 2 [2], we study multi-player concurrent games under imperfect information where (i) $Player_0$'s objective is to reach a target W, and (ii) the opponents are trying to stop this but have partial observation about $Player_0$'s actions. We study the problem of deciding whether the opponents can prevent $Player_0$ to reach W, by beating every $Player_0$'s strategy. We show, using an automata-theoretic approach that, assuming the opponents have the same partial observation and play under uniformity, the problem is in EXPTIME.

In Chapter 3 [3], we introduce and study *Graded Strategy Logic* (GSL), an extension of *Strategy Logic* (SL) with *graded quantifiers*. In GSL, by means of the existential construct $\langle\langle x \geq g \rangle\rangle \varphi$, one can enforce that there exist at least g strategies x satisfying φ. Dually, via the universal construct $[\![x < g]\!]\varphi$, one can ensure that all but less than g strategies x satisfy φ. Strategies in GSL are counted semantically. This means that strategies inducing the same outcome, even though looking different, are counted as one. While this interpretation is natural, it requires a suitable machinery to allow for such a counting, as we do. Precisely, we introduce a non-trivial equivalence relation over strategy profiles based on the strategic behavior they induce. To give an evidence of GSL usability, we investigate some basic questions about the *Vanilla* GSL[1G] fragment, that is the vanilla restriction of the well-studied *One-Goal Strategy Logic* fragment of SL augmented with graded strategy quantifiers. We show that the model-checking problem for this logic is PTIME-COMPLETE. We also report on some positive results about the determinacy.

[1] It appears in [MMS17a].

[2] It appears in [MMS17b].

[3] It appears in [MMMS17].

In Chapter 4 [4], we introduce Graded Strategy Logic (GRADEDSL), an extension of SL by graded quantifiers over tuples of strategy variables, i.e., "there exist at least g different tuples $(x_1, ..., x_n)$ of strategies" where g is a cardinal from the set $\mathbb{N} \cup \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$. We prove that the model-checking problem of GRADEDSL is decidable. We then turn to the complexity of fragments of GRADEDSL. When the g's are restricted to finite cardinals, written $\text{GRADED}_{\mathbb{N}}\text{SL}$, the complexity of model-checking is no harder than for SL, i.e., it is non-elementary in the quantifier rank. We illustrate our formalism by showing how to count the number of different strategy profiles that are Nash equilibria (NE). By analyzing the structure of the specific formulas involved, we conclude that the important problems of checking for the existence of a unique NE can be solved in 2EXPTIME, which is not harder than merely checking for the existence of such an equilibrium.

In Chapter 5 [5], we adopt the view of bounded rationality, and look only at "simple" strategies in specifications of agents' abilities. We formally define what "simple" means, and propose a variant of alternating-time temporal logic that takes only such strategies into account. We also study the model checking problem for the resulting semantics of ability. More precisely, we introduce NatATL*, a logic that extends ATL* by replacing the strategic operator $\langle\langle A \rangle\rangle \varphi$ with a bounded version $\langle\langle A \rangle\rangle^{\leq k} \varphi$, where $k \in \mathbb{N}$ denotes the complexity bound. To measure the complexity of strategies, we assume that they are represented by lists of guarded actions. For memoryless strategies, guards are boolean propositional formulas. For strategies with recall, guards are given as regular expressions over boolean propositional formulas. As technical results, we study the problem of model checking NatATL for both memoryless and memoryfull strategies. The complexity ranges from $\mathbf{\Delta_2^P}$ to $\mathbf{\Delta_3^P}$ in the general case, and from PTIME to $\mathbf{\Delta_2^P}$ for small complexity bounds.

For the sake of clarity of exposition, every chapter is a build in a way that is self content. This means that introduction and preliminary concepts and notations are locally defined.

[4] It appears in [AMMR17].

[5] It appears in [JMM17].

Part I

Reachability Games

CHAPTER 1

Additional Winning Strategies in Reachability Games

Contents

1.1 Introduction

In this chapter, we address the quantitative question of checking whether $Player_0$ has more than a strategy to win a finite two-player game G. We investigate this problem under the *reachability* objective, *i.e.* some states of the game arena are declared *target*. We consider both the cases in which the players have perfect or imperfect information about the moves performed by their opponent. We solve the addressed problem by using an automata-theoretic approach. Precisely, we build an automaton that accepts only trees that are witnesses of more than one winning strategy for the designed player over the game G. Hence, we reduce the addressed quantitative question to the emptiness of this automaton. Our automata solution mainly consists in extending the classic approaches by further individuating a place where $Player_0$ has the ability to follow two different ways (i.e., strategies) to reach a target state. While this may look simple in the perfect information setting, in the imperfect case it requires some careful thoughts. Furthermore, in support to the technical contribution of our solution we observe the following: *(i)* it is an use of automata and an extension of previous approaches never explored before; *(ii)* it provides an elegant solution ; *(iii)* it is an optimal solution as it gives a tight upper bound, *(iv)* it is an easy scalable solution, as one can easily inject more sophisticated solution concepts such as safety, fairness, etc.

By means of the automata-theoretic approach, one can also check for other "forms" of additional winning conditions. For example one can check whether $Player_0$ can win against *all but one* $Player_1$ *strategies*. This is intimately related to the concept of *almost surely-winning* in probabilistic games [ACY95]. As a practical application, this is useful in game design as it can highlight the presence of a unique undesired behavior of the adversarial player and possibly suggest a way to prevent it. Similarly, it is useful in security; for example it can highlight a flow in a firewall (a successful attack coming from the environment) and suggest a way to correct it. Technically, the solution to the question "Does $Player_0$ beat all $Player_1$ strategies but one?" reduces to first build an automaton that collects all tree strategies for $Player_0$ that, except for one path, they correspond to winning strategies, and then check for its non-emptiness.

In a broader vision, the importance of our work resides on the fact that it can be seen as a core engine and as a first step through the efficient solution of important problems in computer science and AI. Among the others, we mention checking the uniqueness of Nash Equilibrium under imperfect information for reachability targets. This field has received much attention recently and some results can be found in top venues such as [BMMRV17, BLMR17]. However, all the approaches used in the mentioned papers lead to a non-elementary complexity, as they are shaped for very reach strategic formalisms to represent the solution concepts, and thus far beyond the tight complexity we achieve instead in this work.

Along the chapter we make use of some cooperative and adversarial game examples

that will help to better explain the specific game setting we are studying and the solution approaches we provide.

Related works. Counting strategies has been deeply exploited in the formal verification of *reactive* systems by means of specification logics extended with *graded modalities*, interpreted over games of infinite duration [BLMV08, FMP08, MMMS15, AMMR16]. However, our work is the first to consider additional winning strategies in the imperfect information setting. Also, it is worth recalling that, on the perfect information side, the solution algorithms present in the literature for graded modalities [MMMS15, AMMR16] have been conceived to address complicated scenarios and, consequently, they usually perform much worse (*w.r.t.* the asymptotic complexity) than our algorithm on the restricted setting we consider. Clearly one can express with graded modalities the existence of additional strategies in a game. To see how this is possible we refer to [AMMR16]for an example in the perfect information setting.

Graded modalities have been first investigated over *closed* systems, i.e., one-player games, to count moves and paths in system models. A pioneering work is [Fin72], where these modalities have been studied in classic modal logic. Successively, they have been exported to the field of *knowledge representation*, to allow quantitative bounds on the set of individuals satisfying specific properties, as well as they have been investigated in first-order logic and description logic. Specifically, they are known as *counting quantifiers* in first-order logics [GOR97], *number restrictions* in *description logics* [HS04, CGLV10, CEO14, BBL15] and *numerical constraints* in query languages [BBL15, FE15]

In [KSV02], *graded* μCALCULUS has been introduced in order to express and evaluate statements about a given number of immediately accessible worlds. Successively in [BMM12], the notion of graded modalities have been extended to deal with number of paths. Among the others graded CTL (GCTL, for short) has been introduced with a suitable axiomatization of counting [BMM12]. That work has been recently extended in [AMR15] to address GCTL*, a graded extension of CTL*.

In this work we analyze and compare different strategies in two-player games. The comparison between strategies is a problematic that has been intensively investigated in other works. Among the others, we mention [JDW02], where the concept of *permissive strategies* has been introduced. However, the aim of that paper is to compare strategies in order to come up with a single strategy that allows to represent all of them by one.

In multi-player system verification, we also witness several specific approaches to count strategies. Chronologically, we first mention *module checking for graded* μCALCULUS [FMP08], where the counting is restricted to moves in a two-player setting. Then, in [MMMS15, AMMR16], motivated by counting Nash equilibria, two different graded extension of Strategy Logic have been considered.

We finally remark that the automata-theoretic solution we provide takes inspiration from the ones used in [KV97, FMP08, BMM12, Tho90, KV00]. In details, in [KV97] such a

technique is used to show that the problem is EXPTIME-complete *w.r.t.* CTL formulas and 2EXPTIME-complete *w.r.t.* CTL* formulas. In [FMP08], an automata-theoretic approach is used to show that the same problem over pushdown structures and graded μCALCULUS formulas is 2EXPTIME-complete. In [BMM12], automata are used to show that graded CTL formulas are satisfiable in exponential time. In [Tho90] efficient algorithms for the emptiness problem of word and tree automata are provided. Finally, in [KV00], alternating tree automata are used in the imperfect information case on the synthesis problem. However, our solution is much more efficient since it is directly constructed for the simpler setting of two-player turn-based games of finite duration, played with respect to the reachability objective.

1.2 Preliminaries

In this section we introduce some preliminary concepts needed to properly define the game setting under exam as well as to describe the adopted solution approach. In particular, we introduce trees useful to represent strategies and automata to collect winning strategies.

Trees. Let Υ be a set. An Υ-*tree* is a prefix closed subset $T \subseteq \Upsilon^*$. The elements of T are called *nodes* and the empty word ε is the *root* of T. For $v \in T$, the set of *children* of v (in T) is $child(T, v) = \{v \cdot x \in T \mid x \in \Upsilon\}$. Given a node $v = y \cdot x$, with $y \in \Upsilon^*$ and $x \in \Upsilon$, we define $anc(v)$ to be y, *i.e.*, the ancestors of v, and $last(v)$ to be x. We also say that v *corresponds* to x. The complete Υ-tree is the tree Υ^*. For $v \in T$, a (full) path π of T from v is a *minimal* set $\pi \subseteq T$ such that $v \in \pi$ and for each $v' \in \pi$ such that $child(T, v') \neq \emptyset$, there is exactly one node in $child(T, v')$ belonging to π. Note that every word $w \in \Upsilon^*$ can be thought of as a path in the tree Υ^*, namely the path containing all the prefixes of w. For an alphabet Σ, a Σ-labeled Υ-tree is a pair $< T, V >$ where T is an $\Upsilon-$tree and $V : T \to \Sigma$ maps each node of T to a symbol in Σ.

Automata Theory. We now recall the definition of *alternating tree automata* and its special case of *nondeterministic tree automata*[VW86, KVW00, EJ91].

Definition 1.2.1 *An* alternating tree automaton *(ATA, for short) is a tuple* $A =< \Sigma, D, Q, q_0, \delta, F >$*, where* Σ *is the alphabet,* D *is a finite set of directions,* Q *is the set of states,* $q_0 \in Q$ *is the initial state,* $\delta : Q \times \Sigma \to \mathcal{B}^+(D \times Q)$ *is the transition function, where* $\mathcal{B}^+(D \times Q)$ *is the set of all positive Boolean combinations of pairs* (d, q) *with* d *direction and* q *state, and* $F \subseteq Q$ *is the set of the accepting states.*

An *ATA* A recognizes (finite) trees by means of (finite) runs. For a Σ-labeled tree $< T, V >$, with $T = D^*$, a run is a $(D^* \times Q)$-labeled N-tree $< T_r, r >$ such that the root

is labeled with (ε, q_0) and the labels of each node and its successors satisfy the transition relation.

For example, assume that A, being in a state q, is reading a node x of the input tree labeled by ξ. Assume also that $\delta(q, \xi) = ((0, q_1) \vee (1, q_2)) \wedge (1, q_1)$. Then, there are two ways along which the construction of the run can proceed. In the first option, one copy of the automaton proceeds in direction 0 to state q_1 and one copy proceeds in direction 1 to state q_1. In the second option, two copies of A proceed in direction 1, one to state q_1 and the other to state q_2. Hence, $\vee$ and $\wedge$ in $\delta(q, \xi)$ represent, respectively, choice and concurrency. A run is *accepting* if all its leaves are labeled with accepting states. An input tree is accepted if there exists a corresponding accepting run. By $L(A)$ we denote the set of trees accepted by A. We say that A is not empty if $L(A) \neq \emptyset$.

As a special case of alternating tree automata, we consider *nondeterministic tree automata* (NTA, for short), where the concurrency feature is not allowed. That is, whenever the automaton visits a node x of the input tree, it sends to each successor (direction) of x at most one copy of itself. More formally, an NTA is an ATA in which δ is in disjunctive normal form, and in each conjunctive clause every direction appears at most once.

1.3 Case Studies

In this section we introduce two different case studies of two-player games. In the first case the players behave adversarial. In the second one, they are cooperative. These running examples are useful to better understand some technical parts of our work.

1.3.1 Cop and Robber Game.

Assume we have a maze where a cop aims to catch a robber, while the latter, playing adversarial, aims for the opposite. For simplicity, we assume the maze to be a grid divided in rooms, each of them named by its coordinates in the plane (see Figure 1.1). Each room can have one or more doors that allow the robber and the cop to move from one room to another. Each door has associated a direction along with it can be crossed. Both the cop and the robber can enter in every room. The cop, being in a room, can physically block only one of its doors or can move in another room. The robber can move in another room if there is a non-blocked door he can take, placed between the two rooms, with the right direction. The robber wins the game if he can reach one of the safe places (EXIT) situated in the four corners of the maze. Otherwise, the robber is blocked in a room or he can never reach a safe place, and thus the cop wins the game. We assume that both the cop and the robber are initially siting in the middle of the maze, that is in the room $(1, 1)$. It important to note that the game is played in a turn-based manner, and the cop is the first player that can moves. Starting from the maze depicted in Figure 1.1, one can see that the robber has only one strategy to win the game. In

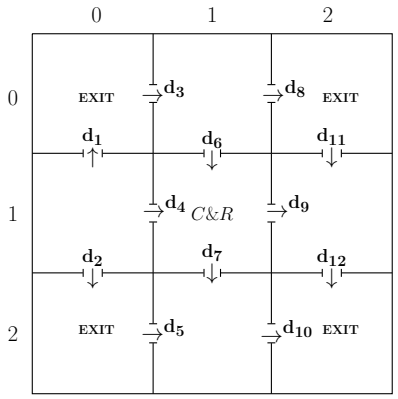


Figure 1.1: Cop and Robber Game.

fact, if the cop blocks the door d_7 (*resp.*, d_9), the robber can choose the door d_9 (*resp.*, d_7), then the cop can go in the room $(1, 2)$ (*resp.*, $(2, 1)$), and finally the robber can choose the door d_{12} (*resp.*, d_{10}) and then wins the game. Consider now two orthogonal variations of the maze. For the first one, consider flipping the direction of the door d_{12}. In this case, the robber loses the game. As second variation, consider flipping the direction of the door d_6. Then the robber wins the game and he has now two strategies to accomplish it.

1.3.2 Escape Game.

Assume we have an arena similar to the one described in the previous example, but now with a cooperative interaction between two players, a human and a controller, aiming at the same target. Precisely, consider the arena depicted in Figure 1.2 representing a building where a fire is occurring. The building consists of rooms and, as before, each room has one-way doors and its position is determined by its coordinates. We assume that there is only one exit in the corner $(2, 2)$. One can think of this game as a simplified version of an automatic control station that starts working after an alarm fire occurs and all doors have been closed. Accordingly, we assume that the two players play in turn and at the starting moment all doors are closed. At each control turn, he opens one door of the room in which the human is staying. The human turn consists of taking one of the doors left open if its direction is in accordance with the move. We assume that there is no communication between the players. We start the game with the human siting in the room $(0, 0)$ and the controller moving first. It is not hard to see that the human can reach the exit trough the doors d_1, d_4, d_7, d_{10} opened by the controller. Actually, this is the only possible way the human has to reach the exit. Conversely, if we consider the scenario in which the direction of the door d_3 is flipped, then there are two strategies to let the human to reach the exit. Therefore, the latter scenario can be considered as better (i.e., more robust) than the former. Clearly, this extra information can be used to

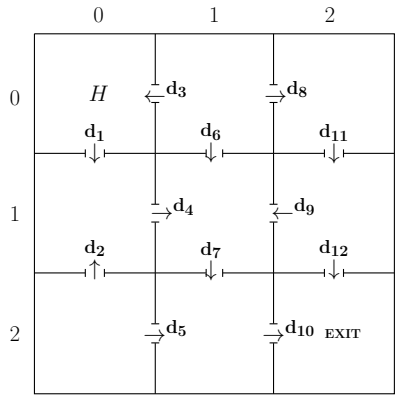


Figure 1.2: Escape Game.

improve an exit fire plan at its designing level.

1.4 The Game Model

In this section, we consider two-player turn-based games that are suitable to represent the case studies we have introduced in the previous section. Precisely, we consider games consisting of an arena and a target. The arena describes the configurations of the game through a set of states, being partitioned between the two players. In each state, only the player that owns it can take a move. This kind of interaction is also known as token-passing. About the target, we consider the reachability objective, that is some states are declared *target*. The formal definition of the considered game model follows.

Definition 1.4.1 *A* turn-based two-player reachability game *(2TRG, for short), played between $Player_0$ and $Player_1$, is a tuple $G \triangleq <\mathrm{St},\ s_I,\ \mathrm{Ac},\ \mathsf{tr},\ \mathrm{W}>$, where $\mathrm{St} \triangleq \mathrm{St_0} \cup \mathrm{St_1}$ is a finite non-empty set of* states*, with St_i being the set of states of $Player_i$, $s_I \in \mathrm{St}$ is a designated* initial state*, $\mathrm{Ac} \triangleq \mathrm{Ac}_0 \cup \mathrm{Ac}_1$ is the set of actions, W is a set of target states, and $\mathsf{tr} : \mathrm{St}_i \times \mathrm{Ac}_i \to \mathrm{St}_{1-i}$, for $i \in \{0,1\}$ is a* transition function *mapping a state of a player and its action to a state belonging to the other player.*

To give the semantics of $2TRG$s, we now introduce some basic concepts such as track, strategy and play. Intuitively, tracks are legal sequences of reachable states in a game that can be seen as descriptions of possible outcomes of the game.

Definition 1.4.2 *A* track *is a finite sequence of states $\rho \in \mathrm{St}^*$ such that, for all $i \in [0, |\rho - 1|[$, if $(\rho)_i \in \mathrm{St_0}$ then there exists an action $a_0 \in \mathrm{Ac}_0$ such that $(\rho)_{i+1} = \mathsf{tr}((\rho)_i, a_0)$, else there exists an action $a_1 \in \mathrm{Ac}_1$ such that $(\rho)_{i+1} = \mathsf{tr}((\rho)_i, a_1)$, where $(\rho)_i$ denotes the i-st element of ρ. For a track ρ, by $last(\rho)$ we denote the last element of ρ and by $\rho_{\leq i}$ we*

denote the prefix track $(\rho)_0 \dots (\rho)_i$. *By* $\mathrm{Trk} \subseteq \mathrm{St}^*$, *we denote the set of tracks over* St. *By* Trk_i *we denote the set of tracks* ρ *in which* $last(\rho) \in \mathrm{St}_i$. *For simplicity, we assume that* Trk *contains only tracks starting at the initial state* $s_I \in \mathrm{St}$.

A *strategy* represents a scheme for a player containing a precise choice of actions along an interaction with the other player. It is given as a function over tracks. The formal definition follows.

Definition 1.4.3 *A* strategy *for* $Player_i$ *in a* $2TRG$ G *is a function* $\sigma_i : \mathrm{Trk}_i \to \mathrm{Ac}_i$ *that maps a track to an action.*

The composition of strategies, one for each player in the game, induces a computation called *play*. More precisely, assume $Player_0$ and $Player_1$ take strategies σ_0 and σ_1, respectively. Their composition *induces* a play ρ such that $(\rho)_0 = s_I$ and for each $i \geq 0$ if $(\rho)_i \in \mathrm{St_o}$ then $(\rho)_{i+1} = \mathsf{tr}((\rho)_i, \sigma_0(\rho_{\leq i}))$, else $(\rho)_{i+1} = \mathsf{tr}((\rho)_i, \sigma_1(\rho_{\leq i}))$.

A strategy is winning for a player if all the plays induced by composing such a strategies with strategies from the adversarial player will enter a target state. If such a winning strategy exists we say that the player wins the game. Reachability games under perfect information are know to be *zero-sum*, *i.e.*, if $Player_0$ loses the game then $Player_1$ wins it and vice versa. The formal definition of *reachability winning condition* follows.

Definition 1.4.4 *Let* G *be a* $2TRG$ *and* $\mathrm{W} \subseteq \mathrm{St}$ *a set of target states.* $Player_0$ *wins the game* G, *under the reachability condition, if he has a strategy such that for all strategies of* $Player_1$ *the resulting induced play will enter a state in* W.

It is folklore that turn-based two-player reachability games are positional [Tho90]. We recall that a strategy is positional if the moves of a player over a play only depends of the last state and a game is positional if positional strategies suffices to decide weather $Player_0$ wins the game. Directly from this result, the following corollary holds.

Corollary 1.4.1 *Given a* $2TRG$ G, *a strategy* σ_0 *for* $Player_0$, *and a strategy* σ_1 *for* $Player_1$, *the induced play* ρ *is winning for* $Player_0$ *if there is* $(\rho)_i \in \mathrm{W}$ *with* $0 \leq i \leq |\mathrm{St}| - 1$.

Hence, the corollary above just states that given a $2TRG$ game, $Player_0$ wins the game if he can reach a winning state in a number of steps bounded by the size of the set of states of the game.

Example 1.4.1 *The two case studies that we have analyzed in Section 1.3 can be easily modeled using a* $2TRG$. *We now give some details. As set of states we use all the rooms in the maze, together with the status of their doors.*

In the Escape Game *the state* $((0,0), \{d_1^c, d_3^c\})$ *is the initial state, where* d_i^c *means that the door* d_i *is closed. For an open door, instead, we will use the label* o *in place of* c. *Formally, let*

$D_{i,j}$ be the set of doors (up to four) belonging to the room (i, j), which can be flagged either with c (closed) or o (open), then we set $\mathrm{St} \subseteq \{((i, j), D_{i,j}) \mid 0 \leq i, j \leq 2\}$. *The set of actions for the controller are* $\mathrm{Ac}_{con} = \{open_{d_i} \mid 0 \leq i \leq 12\}$, i.e. *he chooses a door to open. The set of actions for the human are* $\mathrm{Ac}_{hum} = \{take_{d_i} \mid 0 \leq i \leq 12\}$, i.e. *he chooses a door to take. Transitions are taken by the human in order to change the room (coordinates) or by the controller to change the status of doors. These moves are taken in accordance with the shape of the maze. The partitioning of the states between the players follows immediately, as well as the definition of the target states. A possible track in which the human reaches the exit is* $\rho = ((0,0),\{d_1^c, d_3^c\})((0,0),\{d_1^o, d_3^c\})((0,1),\{d_1^o, d_2^c, d_4^c\})((0,1),\{d_1^o, d_2^c, d_4^o\})((1,1),\{d_4^o, d_6^c, d_7^c, d_9^c\})((1,1),\{d_4^o, d_6^c, d_7^o, d_9^c\})((1,2),\{d_5^c, d_7^o, d_{10}^c\})((1,2),\{d_5^c, d_7^o, d_{10}^o\})((2,2),\{d_{10}^o, d_{12}^c\})$.

In the same way, in Cop and Robber Game *the initial state is* $((1,1),\emptyset)$, *where* $\emptyset$ *means that all doors are open. The set of actions for the cop are* $\mathrm{Ac}_{cop} = \{block_{d_i} \mid 0 \leq i \leq 12\}$, i.e. *he chooses a door to block. The set of actions for the robber are* $\mathrm{Ac}_{rob} = \{take_{d_i} \mid 0 \leq i \leq 12\}$, i.e. *he chooses a door to take.*

1.5 Searching for Additional Winning Strategies

To check whether $Player_0$ has a winning strategy in a $2TRG$ G one can use a classic backward algorithm. We briefly recall it. Let $succ : \mathrm{St} \to 2^{\mathrm{St}}$ be the function that for each state $s \in \mathrm{St}$ in G gives the set of its successors. The algorithm starts from a set S equal to W. Iteratively, it tries to increase S by adding all states $s \in \mathrm{St}$ that satisfy the following conditions: (i) $s \in \mathrm{St}_0$ and $succ(s) \cap S \neq \emptyset$; or, (ii) $s \in \mathrm{St}_1$ and $succ(s) \subseteq S$. If S contains at a certain point the initial state, then $Player_0$ wins the game.

In case one wants to ensure that more than a winning strategy exists, the above algorithm becomes less appropriate. For this reason, we use instead a top-down automata-theoretic approach. To properly introduce this solution we first need to provide some auxiliary notation. Precisely, we introduce the concepts of *decision tree*, *strategy tree*, and *additional strategy tree*.

A decision tree simply collects all the tracks that come out from the interplays between the players. In other words, a decision tree can be seen as an unwinding of the game structure along with all possible combinations of players actions. The formal definition follows.

Definition 1.5.1 *Given a* $2TRG$ G, *a* decision tree *is an* St*-labeled* Ac*-tree* $< T, V >$, *where* ε *is the root of* T, $V(\varepsilon) = s_I$, *and for all* $v \in T$ *we have that:*

- *if* $last(v) \in \mathrm{Ac}_0$ *then* $last(anc(v)) \in \mathrm{Ac}_1$, *otherwise* $last(v) \in \mathrm{Ac}_1$;
- $V(v) = \mathsf{tr}(V(anc(v)), last(v))$.

We now introduce strategy trees that allow to collect, for each fixed strategy for $Player_i$, all possible responding strategies for $Player_{1-i}$, with $i \in \{0, 1\}$. Therefore, the strategy tree

is a tree where each node labeled with $s \in \mathrm{St}_i$ has an unique successor determined by the strategy for $Player_i$ and each node labeled with $s \in \mathrm{St}_{1-i}$ has $|\mathrm{Ac}_{1-i}|$ successors. Thus, a strategy tree is an opportune projection of the decision tree. The formal definition follows.

Definition 1.5.2 *Given a 2TRG and a strategy σ for $Player_i$, a* strategy tree *for $Player_i$ is an* St*-labeled* Ac*-tree $<T, V>$, where ε is the root of T, $V(\varepsilon) = s_I$, and for all $v \in T$ we have that:*

- *if $last(v) \in \mathrm{Ac}_0$ then $last(anc(v)) \in \mathrm{Ac}_1$, otherwise $last(v) \in \mathrm{Ac}_1$;*
- *if $V(anc(v)) \in \mathrm{St}_i$ then $V(v) = \mathsf{tr}(V(anc(v)), \sigma(\rho))$, otherwise $V(v) = \mathsf{tr}(V(anc(v)), last(v))$;*

where $\rho = (\rho)_0 \ldots (\rho)_{|v|-1}$ is a track from s_I, with $(\rho)_k = V(v_{\leq k})$ for each $0 \leq k \leq |v| - 1$.

Following the above definition and Definition 1.4.4, given a $2TRG$ G with a set of target states W, if G is determined then $Player_0$ wins the game and $Player_1$ loses it by simply checking the existence of a strategy tree for $Player_0$, that is a tree such that each path enters a state belonging to W. Such a tree is called a *winning-strategy tree* for $Player_0$.

In case we want to ensure that at least two winning strategies exist then, at a certain point along the tree, $Player_0$ must take two successors. We build a tree automaton that accepts exactly this kind of trees. We now give a definition of additional strategy trees and then we define the desired tree automata.

Definition 1.5.3 *Given a 2TRG G and two strategies σ_1 and σ_2 for $Player_i$, an* additional strategy tree *for $Player_i$ is an* St*-labeled* Ac*-tree $< T, V >$ that satisfies the following properties:*

- *the root node is labeled with the initial state s_I of G;*
- *for each $x \in T$ that is not a leaf and it is labeled with state s of $Player_0$, it holds that x has as children a non-empty subset of $succ(s)$;*
- *for each $x \in T$ that is not a leaf and it is labeled with state s of $Player_1$, it holds that x has as children the set of $succ(s)$;*
- *each leaf of T corresponds to a target state in G;*
- *there exists at least one leaf in T that has an ancestor node x that corresponds to a $Player_0$ state in G and it has at least two children.*

The above definition, but the last item, is the classical characterization of strategy tree. The last property further ensures that $Player_0$ has the ability to enforce at least two winning strategies no matter how $Player_1$ acts.

We now give the main result of this section, *i.e.* we show that it is possible to decide in linear time whether, in a $2TRG$, $Player_0$ has more than a winning strategy. We later report on the application of this result along the case studies.

Theorem 1.5.1 *For a $2TRG$ game G it is possible to decide in linear time whether $Player_0$ has more than a strategy to win the game.*

*Proof.*Consider a $2TRG$ game G. We build an NTA A that accepts all trees that are witnesses of more than a winning strategy for $Player_0$ over G. We describe the automaton. It uses $\mathrm{Q} = \mathrm{St} \times \{ok, split\}$ as set of states where ok and $split$ are flags and the latter is used to remember that along the tree $Player_0$ has to ensure the existence of two winning strategies by opportunely choosing a point where to "split". We set as alphabet $\Sigma = \mathrm{St}$ and initial state $q_0 = (s_I, split)$. For the transitions, starting from a state $q = (s, flag)$ and reading the symbol a, we have that:

$$\delta(q,a) = \begin{cases} (s', ok) & \text{if } s = a \text{ and } s \in \mathrm{St_o} \text{ and } flag = ok; \\ ((s', ok) \wedge (s'', ok)) \vee (s', split) & \text{if } s = a \text{ and } s \in \mathrm{St_o} \text{ and } flag = split; \\ (s_1, ok) \wedge \cdots \wedge (s_n, ok) & \text{if } s = a \text{ and } s \in \mathrm{St_1} \text{ and } flag = ok; \\ (s_1, f_1) \wedge \cdots \wedge (s_n, f_n) & \text{if } s = a \text{ and } s \in \mathrm{St_1} \text{ and } flag = split; \\ \emptyset & \text{otherwise.} \end{cases}$$

where $s', s'' \in succ(s)$ with $s' \neq s''$, $\{s_1, \ldots, s_n\} = succ(s)$, and $f_1, \ldots, f_n$ are flags in which there exists $1 \leq i \leq n$ such that $f_i = split$ and for all $j \neq i$, we have $f_j = ok$. Informally, given a state q, if q belongs to $Player_0$ and its flag is $split$ then there are one successor with flag $split$ or two successors with flag ok. Instead, if the flag is ok then there is only one successor with flag ok. In the case in which the state belongs to $Player_1$ and its flag is ok then there are n successors with flag ok. Finally, if the flag is $split$ then there are $n - 1$ successors with flag ok and one successor with flag $split$.

The set of accepting states is $\mathrm{W} \times \{ok\}$. A tree is accepted by A if all the branches lead to a target state and there is a node labeled with a state in $\mathrm{St_o}$ that has at least two successors. By Corollary 1.4.1, we have that A considers only trees with depth until the number of states, so if no state in W is reached in $|\mathrm{St}|$ steps, then there is a loop over the states in the game model that forbids to reach states in W.

The size of the automaton is just linear in the size of the game. Moreover, by using the fact that, from [Tho90], checking the emptiness of an NTA can be performed in linear time, the desired complexity result follows. □

Example 1.5.1 *Consider the Escape Game example. By applying the above construction, the automaton A accepts an empty language. Indeed, for each input tree, A always leads to a leaf containing either a state with a non-target component (i.e., the tree is a witness of a losing strategy) or with a flag $split$ (i.e., $Player_0$ cannot select two winning strategies). Conversely, consider the same game, but flipping the direction of the door d_3 in the maze. In this case, A is not empty. Indeed, starting from the initial state $(((0,0),\{d_1^c,d_3^c\}),split)$, A proceeds in two different direction with states $(((0,0),\{d_1^o,d_3^c\}),ok)$ and $(((0,0),\{d_1^c,d_3^o\}),ok)$, that refer to two distinct winning strategies for the controller.*

A similar reasoning can be exploited with the Cop and Robber Game example. Indeed, by applying our solution technique, we end in an automaton that accepts an empty language. Conversely, by flipping the door d_4, the automaton accepts a tree that is witnessing of two different winning strategies each of them going through one of the two doors left unblocked by the cop.

For the sake of completeness, we report that in case of one-player games the problem of checking whether more than a winning strategy exists can be checked in NLOGSPACE. Indeed, it is enough to extend the classic *path reachability algorithm* in a way that we search for two paths leading to the target state. This can be done by just doubling the used logarithmic working space [Sip06].

By means of the automata-theoretic approach, one can also check for other and more sophisticated "forms" of additional winning conditions. For example one can check whether $Player_0$ can win the game in case the opponent player is restricted to use *all but one* strategy. This check can be accomplished by first using a classic backward algorithm, introduced at the beginning of this section, for $Player_1$ and then the automaton introduced in the proof of Theorem 1.5.1, but used to collect all additional strategy trees for $Player_1$. Precisely, if the backward algorithm says that $Player_1$ wins the game and the automaton is empty, then the result holds. Indeed, the satisfaction of both these conditions says that $Player_1$ has one and only one strategy to beat all strategies of $Player_0$. Therefore, by removing this specific strategy, $Player_0$ wins the game. So, the explanation of the concept of *all but one* strategy derives.

Theorem 1.5.2 *For a $2TRG$ game G it is possible to decide in linear time whether $Player_0$ can win G against all but one strategies of $Player_1$.*

1.6 Games with Imperfect Information

In this section, we provide the setting of two-player turn-based finite games with imperfect information. As for the perfect information case, we consider here games along the reachability objective. The main difference with respect to the perfect case is that both players

may not have full information about the moves performed by their opponents. Therefore, there could be cases in which a player has to come to a decision (which move to perform) without knowing exactly in which state he is. More precisely, we assume that the players act uniformly, so they use the same moves over states that are indistinguishable to them. The formal definition of these games follows.

Definition 1.6.1 *A* turn-based two-player reachability game with imperfect information *(2TRGI, for short), played between* $Player_0$ *and* $Player_1$*, is a tuple* $G \triangleq <$ St, s_I, Ac, tr, W, $\cong_0$,$\cong_1>$*, where* St*,* s_I*,* Ac*,* tr*, and* W *are as in* $2TRG$*. Moreover,* $\cong_0$ *and* $\cong_1$ *are two equivalence relations over* Ac*.*

Let $i \in \{0, 1\}$. The intuitive meaning of the equivalence relations is that two actions $a, a' \in \mathrm{Ac}_{1-i}$ such that $a \cong_i a'$ cannot be distinguished by $Player_i$. For this reason, we say that a and a' are *indistinguishable* to $Player_i$. By $[\mathrm{Ac}_i] \subseteq \mathrm{Ac}_i$ we denote the subset of actions that are distinguishable for $Player_{1-i}$. If two actions are indistinguishable then also the reached states are so [1]. A relation $\cong_i$ is said an *identity equivalence* if it holds that $a \cong_i a'$ iff $a = a'$. Note that, a $2TRGI$ has perfect information if the equivalence relations contain only identity relations.

To give the semantics of $2TRGI$s, we now introduce the concept of uniform strategy. A strategy is *uniform* if it adheres on the visibility of the players. To formally define it, we first give the notion of indistinguishability over tracks.

For a $Player_i$ and two tracks $\rho, \rho' \in \mathrm{Trk}$, we say that ρ and ρ' are indistinguishable to $Player_i$ iff $|\rho| = |\rho'| = m$ and for each $k \in \{0, \ldots, m-1\}$ we have that $\overline{\mathsf{tr}}((\rho)_k, (\rho)_{k+1}) \cong_i \overline{\mathsf{tr}}((\rho')_k, (\rho')_{k+1})$, where $\overline{\mathsf{tr}}$ is the function that given two states s and s' returns the action a such that $s' = \mathsf{tr}(s, a)$. Note that $\overline{\mathsf{tr}}$ is well defined since it takes as input successive states coming from real tracks and it returns just one unique action due to the specific definition of tr.

Definition 1.6.2 *A strategy* σ_i *is* uniform *iff for every* $\rho, \rho' \in \mathrm{Trk}$ *that are indistinguishable for* $Player_i$ *we have that* $\sigma(\rho) = \sigma(\rho')$*.*

Thus uniform strategies are based on observable actions. In the rest of the paper we only refer to uniform strategies. We continue by giving the definition of the the semantics of $2TRGI$, *i.e.* how $Player_0$ wins the game.

Definition 1.6.3 *Let* G *be a* $2TRGI$ *and* $\mathrm{W} \subseteq \mathrm{St}$ *a set of target states.* $Player_0$ *wins the game* G*, under the reachability condition, if he has a* uniform strategy *such that for all uniform strategies of* $Player_1$ *the resulting induced play has at least one state in* W*.*

[1] For technical reasons, the indistinguishability over states follows from that one over actions. Thanks to this, the construction of the $2TRGI$ easily follows.

Technically, a uniform strategy can be seen as an opportune mapping, over the decision tree, of a player's "strategy schema" built over the visibility part of the decision tree itself. In other words, the player first makes a decision over a set S of indistinguishable states and then this unique choice is used in the decision tree for each state in S. This makes the decision tree to become *uniform*. It is important to observe, however, that we use memoryfull strategies. This means that in a decision tree, the set S of indistinguishable states resides at the same level. To make this idea more precise, we now formalize the concept of *schema strategy tree* and *uniform strategy tree*.

Definition 1.6.4 *Given a* $2TRGI$ *and a uniform strategy* σ *for* $Player_i$*, a* schema strategy tree *for* $Player_i$ *is a* $\{\top, \bot\}$*-labeled (*$\mathrm{Ac}_i \cup [\mathrm{Ac}_{1-i}]$*)-tree* $<T, V>$*, where* ε *is the root of* T*,* $V(\varepsilon) = s_I$*, and for all* $v \in T$ *we have that:*

- *if* $last(v) \in \mathrm{Ac}_i$ *then* $last(anc(v)) \in [\mathrm{Ac}_{1-i}]$*, otherwise* $last(v) \in [\mathrm{Ac}_{1-i}]$*;*
- *if* $last(v) \in [\mathrm{Ac}_{1-i}]$ *then* $V(v) = \top$ *else if* $last(v) = \sigma(\rho)$ *then* $V(v) = \top$*, otherwise* $V(v) = \bot$*;*

where $\rho = (\rho)_0 \ldots (\rho)_{|v|-1}$ *is a track from* s_I*, with* $(\rho)_k = \mathsf{tr}((\rho)_{k-1}, last(v_{\leq k}))$ *for each* $0 \leq k \leq |v| - 1$.

Thus, in a schema strategy tree the $\top$ label indicates that $Player_i$ selects the corresponding set of visible states in the decision tree and the $\bot$ is used conversely [2]. In particular, the starting node of the game is the root of the schema strategy tree and it is always enabled; all nodes belonging to the adversarial player are always enabled; and one of the successors of $Player_i$ nodes is enabled in accordance with the uniform strategy σ. Straightforwardly, a *uniform strategy tree* is a projection of the decision tree along the schema strategy tree. In the next example we consider an extension of the Cop and Robber Game with imperfect information.

Example 1.6.1 *Consider again the Cop and Robber Game example given in Section 1.3. Assume now, that we change the set of actions for the robber in* $\mathrm{Ac}_{rob} = \{l, r, t, b\}$*, where* l*,* r*,* t*, and* b *represent left, right, top, and bottom, respectively, and that the cop can always choose between two doors to enter, namely* d_1 *and* d_2*. Assume also that the cop can only recognize whether the robber moves horizontally or vertically. In other words, it holds that* $l \cong_{cop} r$ *and* $t \cong_{cop} b$*. Accordingly, the cop has only two uniform moves to perform, one for the first pair and one for the second. This is clearly different from the perfect information case where the cop has instead four moves to possibly catch the robber. More formally, in the imperfect information case, assuming that the robber moves first, we have four possible*

[2] The use of $\top$ and $\bot$ is a classical solution in the automata theoretic approach to disable/enable successors, as it has been done in Module Checking [KVW01] and the like.

evolutions of the schema strategy tree. In all schema the root is labeled with $\top$ *and it has two successors* x *and* y*, both labeled with* $\top$ *and corresponding to the actions* l *and* r*, and,* t *and* b*, possibly performed by the robber, respectively. Moreover,* x *and* y *have both two children. The four schema evolve by respectively placing to the children of* x *and* y *the following four combination of* $\top$ *and* $\bot$*:* $((\top, \bot)(\top, \bot))$*,* $((\top, \bot)(\bot, \top))$*,* $((\bot, \top)(\top, \bot))$*, and* $((\bot, \top)(\bot, \top))$*. For example, the second tuple corresponds to choose action* d_1 *in response to actions* l *and* r *and* d_2 *in response to* t *and* b*; similarly, the mining of the other tuple follows. Directly from this explanation it is no hard to build the corresponding uniform strategy trees.*

1.7 Looking for Additional Winning Strategies in 2TRGI

In this section, we introduce an automaton-theoretic approach to solve $2TRGI$, taking as inspiration those introduced in [KV97, FMP08, BMM12, Tho90]. We start by analyzing the basic case of looking for a winning strategy. We recall that this problem is already investigated and solved in the case in which there is imperfect information over states [Rei84, CDHR07]. By these considerations, we show how to solve the case in which the imperfect information is over the actions. Subsequently, we extend the latter case to check whether the game also admits additional winning strategies.

Before starting we recall that positional strategies do not suffice to decide a game with imperfect information. Indeed, it is well known that $Player_0$ needs exponential memory *w.r.t.* the size of the states of the game in order to come up with a winning strategy in case it exists [CDHR07]. Therefore, we cannot use directly the approach exploited in Section 1.5. A possible direction to solve a game G with imperfect information is to convert it, by means of a subset construction, in a game $\bar{G}$ with perfect information and solve it by using Theorem 1.5.1 (see for example [CDHR07]). With this translation one can individuate along the game $\bar{G}$ exponential strategies necessary to $Player_0$ for winning the game. As the subset construction involves an exponential blow-up and Theorem 1.5.1 provides a polynomial-time solution we get an overall exponential procedure. In this paper, however, we present a different and more elegant way to solve games with imperfect information. Precisely, we introduce a machinery that in polynomial time can represent exponential strategies. With more details, given a game with imperfect information G we construct an alternating tree automaton that accepts trees that represent uniform strategies under imperfect information. This is done by sending the same copy of the automaton (same direction) to all states that are indistinguishable to $Player_0$. Then, the automaton checks that in all these common directions $Player_0$ behaves the same and satisfies the reachability condition. Precisely, the automaton takes in input trees corresponding to $Player_0$'s strategies over the unwinding of the game by replacing nodes by the equivalence classes. The run instead is as usual, that is a $Player_0$ strategy over the total

unwinding of the game. The beauty of this approach resides on the fact that we do not make explicit the exponential strategies required to win the game but rather consider a polynomial compact representation of them by means of the automaton. Clearly, as the emptiness of alternating tree automata is exponential, we get the same overall exponential complexity as in the subset construction approach.

1.7.1 Solution 2TRGI

To solve $2TRGI$, we use an automata-approach via alternating tree automata. The idea is to read a $\{\top, \bot\}$-labeled $(\mathrm{Ac_o} \cup [\mathrm{Ac_1}])$-tree such that more copies of the automaton are sent to the same directions along the class of equivalence over $[\mathrm{Ac_1}]$.

Theorem 1.7.1 *Given a $2TRGI$ G played by $Player_0$ and $Player_1$, the problem of deciding whether $Player_0$ wins the game is* ExpTime-complete.

Proof. Let G be a $2TRGI$. For the lower bound, we recall that deciding the winner in a 2-player turn-based games with imperfect information is ExpTime-hard [Rei84, CDHR07].

For the upper bound, we use an automata-theoretic approach. Precisely, we build an ATA A that accepts all schema strategy trees for $Player_0$ over G. The automaton, therefore will send more copies on the same direction of the input tree when they correspond to hidden actions. Then it will check the consistency with the states on the fly by taking in consideration the information stored in the node of the tree. This can be simply checked by means of a binary counter along with the states of the automaton. For the sake of readability we omit this.

The automaton uses as set of states $Q = \mathrm{St} \times \mathrm{St} \times \{\top, \bot\} \times \{0, 1\}$ and alphabet $\Sigma = \{\top, \bot\}$. Note that, we use in Q a duplication of game states as we want to remember the game state associated to the parent node while traversing the tree. For the initial state we set $q_0 = (s_I, s_I, \top, 0)$, *i.e.*, for simplicity the parent game state associated to the root of the tree is the game state itself. The flag $f \in \{0, 1\}$ indicates whether along a path we have entered a target state, in that case we move f from 0 to 1. Given a state $q = (s, s', t, f)$, the transition relation is defined as:

$$\delta(q, t') = \begin{cases} \bigwedge_{a_0 \in \mathrm{Ac_0}} (d, (s', s'', \top, f')) & \text{if } s' \in \mathrm{St_o} \text{ and } t' = \top \text{ and } t = \top; \\ \bigwedge_{a_1 \in \mathrm{Ac_1}} (d, (s', s'', \top, f')) & \text{if } s' \in \mathrm{St_1} \text{ and } t' = \top \text{ and } t = \top; \\ false & \text{if } t' = \top \text{ and } t = \bot; \\ true & \text{if } t' = \bot. \end{cases}$$

where if $s' \in \mathrm{St_o}$ then $s'' = tr(s', a_0)$ and d is in accordance with $|\mathrm{Ac_1}|$, else $s'' = tr(s', a_1)$ and d is in accordance with $|\mathrm{Ac_0}|$; if $q' \in \mathrm{W}$ then $f' = 1$ otherwise $f' = f$.

Informally, given a state q, if q belongs to $Player_0$ (*resp.*, $Player_1$) and it is enabled then there are $|\mathrm{Ac}_0|$ (*resp.*, $|\mathrm{Ac}_1|$) enabled successors. Instead, if q is disabled then the automaton returns $false$. Finally, if the automaton reads the symbol $\perp$ then it returns $true$.

The set of accepted states is $\mathrm{F} = \{(s, s', t, f) : s, s' \in \mathrm{St} \wedge t = \top \wedge f = 1\}$. Recall that an input tree is accepted if there exists a run whose leaves are all labeled with accepting states. In our setting this means that an input tree simulates a schema strategy tree for $Player_0$. So, if the automaton is not empty then $Player_0$ wins the game, *i.e.*, there exists a uniform strategy for him.

The required computational complexity of the solution follows by considering that: *(i)* the size of the automaton is polynomial in the size of the game, *(ii)* to check its emptiness can be performed in exponential time [EJ88, KVW00]. □

1.7.2 Additional Winning Strategies for 2TRGI

In this section we describe the main result of this work, *i.e.*, we show an elementary solution to ensure that more than a winning strategy exists in $2TRGI$s. As we have anticipated earlier we use an opportune extension of the automata-theoretic approach we have introduced in the previous sections.

First of all, we formalize the concept of *schema additional strategy tree*.

Definition 1.7.1 *Given a $2TRGI$ and two uniform strategies σ and σ' for $Player_i$, a* schema additional strategy tree *for $Player_i$ is a $\{\top, \perp\}$-labeled ($\mathrm{Ac}_i \cup [\mathrm{Ac}_{1-i}]$)-tree $<T, V>$, where ε is the root of T, $V(\varepsilon) = s_I$, and for all $v \in T$ we have that:*

- *if $last(v) \in \mathrm{Ac}_i$ then $last(anc(v)) \in [\mathrm{Ac}_{1-i}]$, otherwise $last(v) \in [\mathrm{Ac}_{1-i}]$;*
- *if $last(v) \in [\mathrm{Ac}_{1-i}]$ then $V(v) = \top$ else if $last(v) = \sigma(\rho)$ or $last(v) = \sigma'(\rho)$ then $V(v) = \top$, otherwise $V(v) = \perp$;*

where $\rho = (\rho)_0 \ldots (\rho)_{|v|-1}$ is a track from s_I, with $(\rho)_k = \mathsf{tr}((\rho)_{k-1}, last(v_{\leq k}))$ for each $0 \leq k \leq |v| - 1$.

Informally, a schema additional strategy tree is a schema strategy tree in which at a certain point along the tree, a state of $Player_0$ has to two successors. We build a tree automaton that accepts exactly this kind of trees. Now, we have all ingredients to give the following result.

Theorem 1.7.2 *Given a $2TRGI$ G played by $Player_0$ and $Player_1$, the problem of deciding whether $Player_0$ has more than a uniform strategy to win the game is in* ExpTime-complete.

Proof. Let G be a $2TRGI$. For the lower bound, we recall that deciding the winner in a 2-player turn-based games with imperfect information is ExpTime-hard [Rei84, CDHR07].

For the upper bound, we use an automata-theoretic approach. Precisely, we build an ATA A that accepts all schema additional strategy trees for $Player_0$ over G. Since the automaton sends more copies on the same direction of the input tree when they correspond to hidden actions, then it checks the consistency with the states on the fly by taking in consideration the information stored in the node of the tree. In detail, the automaton uses as set of states $Q = \mathrm{St} \times \mathrm{St} \times \{\top, \bot\} \times \{0, 1\} \times \{ok, split\}$, where given a state $q = (s, s', t, f, \bar{f})$ we have that s is the parent of s', s' is the actual state, t is used to disable/enable the state, f is a flag indicating whether along the path we have entered in a target state, and $\bar{f}$ is a flag indicating whether along the path there was a state of $Player_0$ with two successors. The alphabet is $\Sigma = \{\top, \bot\}$ and the initial state is $q_0 = (s_I, s_I, \top, 0, split)$. Given a state $q = (s, s', t, f, \bar{f})$, the transition relation is defined as follows:

$$\delta(q, t') =$$

$$\begin{cases} \bigwedge_{a_0 \in \mathrm{Ac_0}} (d, (s', s'', \top, f', ok)) & \text{if } s' \in \mathrm{St_o} \text{ and } t' = \top \text{ and } t = \top \text{ and } \bar{f} = ok; \\ \bigwedge_{a_0 \in \mathrm{Ac_0}} \bigvee_{\bar{f}' \in \{ok, split\}} (d, (s', s'', \top, f', \bar{f}')) & \text{if } s' \in \mathrm{St_o} \text{ and } t' = \top \text{ and } t = \top \text{ and } \bar{f} = split; \\ \bigwedge_{a_1 \in \mathrm{Ac_1}} (d, (s', s'', \top, f', ok)) & \text{if } s' \in \mathrm{St_1} \text{ and } t' = \top \text{ and } t = \top \text{ and } \bar{f} = ok; \\ \bigwedge_{a_1 \in \mathrm{Ac_1}} \bigvee_{\bar{f}' \in \{ok, split\}} (d, (s', s'', \top, f', \bar{f}')) & \text{if } s' \in \mathrm{St_1} \text{ and } t' = \top \text{ and } t = \top \text{ and } \bar{f} = split; \\ false & \text{if } t' = \top \text{ and } t = \bot; \\ true & \text{if } t' = \bot. \end{cases}$$

where it holds that if $s' \in \mathrm{St_o}$ then $s'' = tr(s', a_0)$ and d is in accordance with $|\mathrm{Ac_1}|$, otherwise $s'' = tr(s', a_1)$ and d is in accordance with $|\mathrm{Ac_0}|$; if $q' \in \mathrm{W}$ then $f' = 1$ otherwise $f' = f$. Informally, given a state q, if q belongs to $Player_0$, it is enabled, and its flag is $split$ so there are $|\mathrm{Ac_0}|$ enabled successors, such that it holds that either all of them have $split$ has flag, or at least two of them have ok as flag . Instead, if the state q is enabled and its flag is ok then there are $|\mathrm{Ac_0}|$ enabled successors and all of them have the flag ok. If the state q belongs to $Player_1$, it is enabled, and its flag is ok, thus there are $|\mathrm{Ac_1}|$ enabled successors such that all of them have the flag ok. Instead, if the state is enabled and its flag is $split$ then there are $|\mathrm{Ac_1}|$ enabled successors such that at least one successor has flag $split$. In the case in which the state q is disabled then the automaton returns $false$. Finally, if the automaton reads the symbol $\bot$ then it returns $true$.

The set of accepted states is $\mathrm{F} = \{(s, s', t, f, \bar{f}) : s, s' \in \mathrm{St} \wedge t = \top \wedge f = 1 \wedge \bar{f} = ok\}$. Recall that an input tree is accepted if there exists a run whose leaves are all labeled with accepting states. In our setting this means that an input tree simulates a schema additional strategy tree for $Player_0$. So, if the automaton is not empty then $Player_0$ wins the game, *i.e.*, there exists a schema additional strategy tree for him. The required computational complexity of the solution follows by considering that: *(i)* the size of the automaton is

polynomial in the size of the game, *(ii)* to check its emptiness can be performed in exponential time [EJ88, KVW00]. □

Finally, also in the imperfect information case one can repeat the same reasoning done in Section 1.5 about "all but one" strategies. Indeed, it is sufficient to use the automata in the proofs of Theorem 1.7.1 and Theorem 1.7.2 from the viewpoint of $Player_1$. Indeed, the result follows by checking whether the former automaton is not empty and the latter automaton is empty. Consequently, the following result holds.

Theorem 1.7.3 *For a $2TRGI$ game G it is possible to decide in* EXPTIME-COMPLETE *whether $Player_0$ has a uniform strategy against all but one uniform strategies of $Player_1$.*

1.8 Conclusion and Future Work

In this chapter we have introduced a simple but effective automata-based methodology to check whether a player has more than a winning strategy in a two-player game under the reachability objective. Our approach works with optimal asymptotic complexity both in the case the players have perfect information about the moves performed by their adversarial or not. Overall, this is the first work dealing with the counting of strategies in the imperfect information setting we are aware of.

We have showed how our approach can be applied in practice by reporting on its use over two different game scenarios, one cooperative and one adversarial. We believe that the solution algorithm we have conceived in this chapter can be used as core engine to count strategies in more involved game scenarios and in many solution concepts reasoning. For example, it can be used to solve the *Unique Nash Equilibrium* problem, in an extensive game form.

This work opens to several interesting questions and extensions. An interesting direction is to consider the counting of strategies in multi-agent concurrent games. This kind of games have several interesting applications in artificial intelligence [Woo02]. Some works along this line have been done, but not for finite games, nor in the imperfect information setting. As another direction of work, one can consider some kind of hybrid game, where one can opportunely combine teams of players working concurrently with some others playing in a turn-based manner as in [JM14, JM15, MS15]. Last but not least, it would be worth investigating infinite-state games. These games arise for example in case the interaction among the players behaves in a recursive way [BMP10, MP15].

CHAPTER 2

Hiding actions in Multi-Player Games

Contents

2.1 Introduction

In this chapter we consider multi-player reachability games, played by n players $Player_0 \dots Player_{n-1}$, where $Player_{i>0}$ can have (equal) *imperfect information about the actions* taken by $Player_0$. Conversely, $Player_0$ has always full observability over the actions taken by the other players. Some states of the game arena are set as target and the aim of $Player_1 \dots Player_{n-1}$ is to prevent $Player_0$ from reaching a target state, otherwise $Player_0$ wins the game. Precisely, we check whether $Player_0$ have a counter-strategy to every joint-strategy of his opponents, or equivalently that $Player_1 \dots Player_{n-1}$ do not have a winning strategy. Clearly, all players will act by adhering to their observability. Solving the game amounts to checking whether $Player_0$ wins the game.

The game model we consider can be applied in a number of concrete scenarios. As an example, it can be used in the context of Wireless Sensor Networks [AV10], which consist of a large number of small sensors that are used for gathering data in a variety of environments. The data collected by each sensor is communicated through the network to a single processing center that uses all reported data to determine characteristics of the environment or detect an event. The communication or message passing process is usually designed in a way that it limits the consumption of energy. For this reason, some sensors have a limited scanner view [BC03]. This scenario can be easily casted in our game setting, where the information from sensors can be seen as actions, as well as it is for the processing center who has complete information from the sensors. Then, it is possible to check whether a specific configuration (for example a critical one) can be reached. Other examples can be found in the setting of not losing games [GLLS07].

Deciding the winner of the introduced multi-player concurrent game setting requires addressing a major issue: we have to limit the check to solely those players' strategies that are compatible with the visible information. Note that the visibility constraint is not a property easy to check [KV97]. In particular, an imperfect information at a certain round of the game may propagate along all future interactions stages and this has to be taken into account in every single play. We address this difficulty by introducing an *ad hoc* structure, named *blocking tree*. Precisely, we consider a tree that, at each node and for every possible action taken by $Player_0$, collects the best possible counter-actions of the adversarial players, chosen under the visibility constraint. Such a tree is considered "blocking" whenever it contains only paths along which no target state is met. Then, we say that $Player_0$ wins the game if and only if no blocking tree exists. Otherwise, we say that $Player_0$ loses the game and then the opponents win it. By using this reasoning and by exploiting an automata-theoretic approach we show that deciding our game setting can be done in EXPTIME. Precisely, we build an alternating tree automaton [EJ99] that accepts all blocking trees and reduce the game decision problem to check its emptiness. As the automata construction is linear and checking its emptiness is

exponential, we get the result.

Regarding the automata we use, recall that nondeterministic tree automata, on visiting a node of the input tree, send exactly one copy of themselves to each successor of the node. An alternating automaton instead can send several copies of itself to the same successor. To this purpose, the automaton uses directions to indicate to which successor to send a state. In our setting, we set as directions the product of the common visible actions among all the players. This allows to keep together states that, looking the same to those players, share the same chosen actions. Note that while the input tree has a very "thin shape" due to the imperfect information setting, the corresponding run has as branching degree the product of the actions all players can choose. Also, we have a tight complexity since turn-based 2-player games with imperfect information are EXPTIME-hard [KV97].

Related works. Imperfect information games have been largely considered in the literature [DT11, KV00, JÅ07, Rei84, BJ14]. A seminal work is [Rei84] in which a number of variants of 2-player games with imperfect information have been investigated. Generally, having imperfect information immediately reflects on worsening the complexity of solving the game. In multi-player games one can easily jump to non-elementary [PR89] or even to undecidability [DT11]. As an evidence we mention [PR90] where a pipeline of processes architecture is considered and each output communication of process i is taken as the input communication of process $i + 1$. The reachability problem under imperfect information in this specific setting is decidable but complete for non-elementary time[1]. In [vdMW05] the authors impose a hierarchy in order to regain decidability of the synthesis problem under imperfect information. As in [PR90] the problem is decidable but non-elementary. In contrast, as we discussed in the rest of the chapter, our (automata) procedure is 1-EXPTIME-COMPLETE. Moreover, differently from [vdMW05], we use concurrency, imperfect actions, and specific adversarial coalitions. Other works worth of mention concern ATL*, a logic introduced by Alur, Kupferman and Henzinger [AHK02]. In many cases, deciding the related decision problem becomes undecidable [AHK02, DT11]. In particular it is undecidable in the case of three agents having perfect recall of the past (as we do), while it is elementary decidable in case the agents play positional (a.k.a. memoryless) strategies. However note that in the ATL* setting the agents can learn during a play and possibly move to perfect information about the game structure. This is a very strong requirement that conflicts with several application domains (see [KVW01] for an argument) and we do not consider it here.

A group of works that is closely related to our setting concerns *module checking* [KV97, KVW01, JM14]. In the basic setting this is a two-player game where one of the players, the environment, has nondeterministic strategies. Module checking has been also investigated in the imperfect information setting and the related works have been a good source of inspiration

[1] Other settings have been also taken in consideration and leading to an undecidable problem.

for the solution techniques we propose in this chapter. Note that in module checking the system player ($Player_0$, in our case) has one fixed strategy, while the adversarial environment ($Player_1$) has imperfect information about the arena (and thus the actions performed by $Player_0$). Our work can be seen as a specific multi-player variant of module checking under deterministic strategies.

Other works dealing with imperfect information games and worth of mentioning are [BK10, CDHR07, CH12]. These works consider two-player turned-based games rather than concurrent multi-player arenas, as we do. On the other hand they consider richer structures (such as stochastic arenas) and/or richer winning conditions.

Close to our setting is also the game structure studied in [CD14]. There the authors consider reachability three-player concurrent games under some specific form of imperfect information but with no hierarchy over the visibility of actions. Solving such a game turns out to be non-elementary. Finally we report that, in a short paper recently published, a preliminary study on reachability games under imperfect information have been considered along with a winning condition similar to the one we use here [MMS16]. Our work improves and extends all the results reported there on two-player games and, more important, introduces fresh results on the multi-agent side. For the sake of readability we also use some concepts introduced there.

We conclude this section by remarking that our definition of imperfect information relies on the actions played by the players involved in the game, rather than the visited states This allows to reason about the actions played by other players and not only the outcome of these actions. Apart few works we are aware of [KM98, San07, FS05, CD14], this direction has been less explored in literature, but shown to be useful in several contexts. In particular, in the reasoning about Nash Equilibria, such an extra information plays a key role [AAK15, San07].

2.2 Game Definition

In this section we define the multi-player reachability game under interest as well as some preliminary notions. We consider that $Player_1, \ldots, Player_{n-1}$ can have imperfect information about the actions performed by $Player_0$. Instead, $Player_0$ is omniscient and has perfect information about all other players.

Model. We model the game by means of a classic *concurrent game structure* [AHK02] augmented with a set of *target states* and an *equivalence relation* over $Player_0$'s actions. The formal definition follows.

Definition 2.2.1 *A* concurrent multi-player reachability game with imperfect information *(CRGI, for short) is a tuple* $G \triangleq <\mathrm{St}, s_I, \mathrm{P}, \mathrm{Ac}, \mathsf{tr}, \mathrm{W}, \cong>$ *where* St *is a finite non empty set of* states*,* $s_I \in \mathrm{St}$ *is a designated* initial state*,* $\mathrm{P} \triangleq \{Player_0, \ldots, Player_{n-1}\}$ *is the set of*

players, $\mathrm{Ac} \triangleq \mathrm{Ac}_0 \cup \ldots \cup \mathrm{Ac}_{n-1}$ *is the set of actions. We assume that* $\mathrm{Ac}_i \cap \mathrm{Ac}_j = \emptyset$*, for each* $0 \leq i, j < n$*.* $\mathrm{W} \subseteq \mathrm{St}$ *is a set of target states,* $\mathsf{tr} : \mathrm{St} \times (\mathrm{Ac}_0 \times \ldots \times \mathrm{Ac}_{n-1}) \to \mathrm{St}$ *is a* transition function *mapping a tuple made of a state and one action for each player to a state, and* $\cong$ *is an equivalence relations on* Ac_0*.*

W.l.o.g., we assume that for each pair of states s and s' there exists at most one tuple of players' actions that lets to transit from s to s'. Observe that one can always transform an arbitrary $CRGI$ to make this property true by opportunely duplicating the states that are reachable along different agents' decisions, starting from a common state.

For two actions $a, a' \in \mathrm{Ac}_0$, we say that a and a' are *indistinguishable/invisible* to all players $Player_1, \ldots, Player_{n-1}$ if $a \cong a'$. Moreover, we fix with $[\mathrm{Ac}_0] \subseteq \mathrm{Ac}_0$ as a set of representative actions over $\cong$. If two actions are indistinguishable for a player then also the reached states are so. That is, the imperfect information over actions induces the imperfect information over states. It is important to note that, in our setting all players can distinguish the initial state while this is not true in general, in case of imperfect information over states. A relation $\cong$ is said to be an *identity equivalence* if $a \cong a'$ iff $a = a'$.

A $CRGI$ has perfect information if $\cong$ contains only identity relations (so, we drop I from the acronym). A $CRGI$ is a 2-player game if $\mathrm{P} = \{Player_0, Player_1\}$ and we name it $2CRGI$. Hence, $2CRG$ are 2-player games under perfect information.

Tracks, strategies, and plays. To give the semantics of $CRGI$s, we now introduce some basic concepts such as track, strategy, and play.

Definition 2.2.2 *A* track *is a finite sequence of states* $\rho \in \mathrm{St}^*$ *such that, for all* $i \in [0, |\rho| - 1[$*, there exists* n *actions* $a_0 \in \mathrm{Ac}_0, \ldots, a_{n-1} \in \mathrm{Ac}_{n-1}$ *such that* $(\rho)_{i+1} = \mathsf{tr}((\rho)_i, a_0, \ldots, a_{n-1})$*, where* $(\rho)_i$ *is the* i*th element of* ρ*.*

For a track ρ, by $\rho_{\leq i}$ we denote the prefix track $(\rho)_0 \ldots (\rho)_i$. By $\mathrm{Trk} \subseteq \mathrm{St}^*$, we denote the set of tracks over St. For simplicity, we assume that Trk contains only tracks starting at the initial state $s_I \in \mathrm{St}$.

A *strategy* represents a scheme for a player containing a precise choice of actions along an interaction with the other players. It is given as a function over tracks. The formal definition follows.

Definition 2.2.3 *A* strategy *for* $Player_i$ *in a CRGI* G *is a function* $\sigma_i : \mathrm{Trk} \to \mathrm{Ac}_i$ *mapping each track to an action.*

A strategy is *uniform* if it adheres on the visibility of the players. To formally define it, we first give the notion of indistinguishability over tracks.

Let $\overline{\mathsf{tr}} : \mathrm{St} \times \mathrm{St} \rightharpoonup (\mathrm{Ac}_0 \times \ldots \times \mathrm{Ac}_{n-1})$ a partial function that given two states s and s' returns, if exists, the tuple of actions $a_0, \ldots, a_{n-1}$ such that $s' = \mathsf{tr}(s, a_0, \ldots, a_{n-1})$. Note

that $\overline{\mathsf{tr}}$ is well defined as we assume that for each pair of states s and s' there exists at most one tuple of players' actions that allows us to move from s to s'.

Definition 2.2.4 *Given two tracks $\rho, \rho' \in \mathrm{Trk}$, we say that ρ and ρ' are* indistinguishable *to $Player_j$, with $j > 0$, iff* (i) $|\rho| = |\rho'| = m$; (ii) *for each $k \in \{0, \ldots, m-1\}$ it holds that $\overline{\mathsf{tr}}((\rho)_k, (\rho)_{k+1})(0) \cong \overline{\mathsf{tr}}((\rho')_k, (\rho')_{k+1})(0)$.*

We can now define the concept of uniform strategy.

Definition 2.2.5 *A strategy σ_i is* uniform *iff for every $\rho, \rho' \in \mathrm{Trk}$ that are indistinguishable for $Player_i$ we have that $\sigma_i(\rho) = \sigma_i(\rho')$.*

Thus uniform strategies are based on observable actions. In the rest of the paper we only refer to uniform strategies.

The composition of strategies, one for each player in the game, induces a computation called *play*. More precisely, assume $Player_0, \ldots, Player_{n-1}$ take strategies $\sigma_0, \ldots, \sigma_{n-1}$, respectively. Their composition induces a play ρ such that $(\rho)_0 = s_I$ and for each $i \geq 0$ we have that $(\rho)_{i+1} = \mathsf{tr}((\rho)_i, \sigma_0(\rho_{\leq i}), \ldots, \sigma_{n-1}(\rho_{\leq i}))$, for all $i \in \mathbb{N}$.

Now, we give the concepts towards the definition of the the semantics of $CRGI$, *i.e.* how $Player_0$ wins the game. For a matter of presentation, we reason about the simpler case of $2CRG$. Most of the concepts we present here will be used or opportunely extended to define $CRGI$.

Reachability winning condition. To make our reasoning clear, we recall the classic definition of *reachability winning condition* and then discuss its rule in our game setting. First of all, we define the concept of *winning strategy*.

Definition 2.2.6 *Let G be a $2CRG$ and W a set of target states. A strategy σ is winning for $Player_0$ (resp., $Player_1$) over G under the reachability condition, if for all strategies of $Player_1$ (resp., $Player_0$) the resulting induced plays have at least one (resp., no) state in W.*

In reachability games, if a player has a winning strategy, we say that he wins the game, as reported in the following definition.

Definition 2.2.7 *Let G be a $2CRG$ and W a set of target states. $Player_0$ (resp., $Player_1$) wins the game G under the reachability condition, if he has a winning strategy.*

It is important to observe that the above definition does not guarantee that the game always admits a winner. In fact, there are scenarios in which no one of the players has a winning strategy, that is a strategy that beats all counter strategies of the opponent player. One can be convinced of this by simply considering the classic two-player concurrent *matching bit*

game. Indeed, by applying Definition 2.2.7 we have that neither $Player_0$ wins the game nor $Player_1$ does.

Trees. In this paper we are going to use a different winning condition to establish whether $Player_0$ wins the game. We formalize this condition by means of trees. For this reason we first recall some basic notation about this structure.

Let Υ be a set. An Υ-*tree* is a prefix closed subset $T \subseteq \Upsilon^*$. The elements of T are called *nodes* and the empty word ε is the *root* of T. For $v \in T$, the set of *children* of v (in T) is $child(T, v) = \{v \cdot x \in T \mid x \in \Upsilon\}$. Given a node $v = y \cdot x$, with $y \in \Upsilon^*$ and $x \in \Upsilon$, we define $prf(v)$ to be y and $last(v)$ to be x. We also say that v *corresponds* to x. The complete Υ-tree is the tree Υ^*. For $v \in T$, a (full) path π of T from v is a *minimal* set $\pi \subseteq T$ such that $v \in \pi$ and for each $v' \in \pi$ such that $child(T, v') \neq \emptyset$, there is exactly one node in $child(T, v')$ belonging to π. Note that every word $w \in \Upsilon^*$ can be thought of as a path in the tree Υ^*, namely the path containing all the prefixes of w. For an alphabet Σ, a Σ-labeled Υ-tree is a pair $< T, V >$ where T is an $\Upsilon-$tree and $V : T \to \Sigma$ maps each node of T to a symbol in Σ.

The considered winning condition. In this paper we consider the setting in which $Player_1$ wins the game (and thus $Player_0$ loses it) if, for each strategy of $Player_0$ there exists a strategy for $Player_1$, that can force the induced play to avoid a target state. Otherwise, $Player_0$ wins the game. Under this definition it is immediate to observe that a game always has a winner, under both perfect and imperfect information. The winning condition we adopt simply enforces the winning power of $Player_0$ under imperfect information. However observe that under this condition, we still have cases (in particular in the perfect information setting) in which $Player_1$ does not have a winning strategy but still he can block $Player_0$ and thus the latter loses the game (see Section 2.3 for an example). We formalize our new winning condition by means of a tree structure that we call *blocking tree*. To proper introduce this structure we also need to provide the concepts of *decision tree* and *strategy tree*.

We now give the notion of decision tree. Such a tree simply collects all the tracks that come out from the interplays between the players. In other words, a decision tree can be seen as an unwinding of the game structure along with all possible combinations of player actions. More formally, given a 2*CRG* G, a *decision tree* is an St-labeled full $(\mathrm{Ac}_0 \times \mathrm{Ac}_1)^*$-tree collecting all tracks over G.

We now introduce strategy trees that allow to collect, for each fixed strategy for $Player_i$, all possible responding strategies for $Player_{1-i}$, with $i \in \{0, 1\}$. Therefore, the strategy tree is a full tree whose directions are determined by Ac_{1-i} and it is labeled with states given in accordance with the transition function of the game based on the fixed strategies for $Player_i$ and all possible strategies of $Player_{1-i}$. Thus, a strategy tree is an opportune projection of the decision tree. The formal definition follows.

Definition 2.2.8 *Given a 2CRG G and a strategy σ for $Player_i$, a* strategy tree *for $Player_i$ is an* St*-labeled full Ac_{1-i}^*-tree $< \mathrm{Ac}_{1-i}^*, l >$, with l as follows:*

1. $V(\varepsilon) = s_I$;
2. *for all $v \in \mathrm{Ac}_{1-i}^+$, let $\rho = (\rho)_0 \dots (\rho)_{|v|-1}$ be a track from s_I, with $(\rho)_k = l(v_{\leq k})$ for each $0 \leq k < |v|$. We have that $V(v) = \mathsf{tr}(V(prf(v)), act_0, act_1)$, with $act_i = \sigma(\rho)$ and $act_{1-i} = last(v)$.*

The strategy tree can be used to check whether a strategy is winning in accordance to Definition 2.2.6. In fact, given a $2CRG$ G with a set of target states W, then $Player_0$ wins the game under the reachability condition by simply checking the existence of a strategy tree for $Player_0$, that is a tree such that each path enters a state belonging to W. Such a tree is called a *winning-strategy tree* for $Player_0$. Analogously, one can check whether $Player_0$ cannot win the game by checking whether $Player_1$ can "block" every possible strategy for $Player_0$. This blocking behavior that let $Player_0$ losing the game can be collected in a blocking tree for $Player_0$. The definition of blocking tree follows.

Definition 2.2.9 *Given a 2CRG G a* blocking tree *for $Player_i$ is a $\{\top, \bot\}$-labeled $(\mathrm{Ac}_0 \times \mathrm{Ac}_1)$-tree $< T, V >$ with $T \subset (\mathrm{Ac}_0 \times \mathrm{Ac}_1)^*$ and V as follows:*

1. $V(\varepsilon) = \top$;
2. *for all nodes $v \in T$, we have that $\rho = (\rho)_0 \dots (\rho)_{|v|-1}$ is a track from s_I such that for each $0 < k < |v|$ it holds that $(\rho)_k = tr((\rho)_{k-1}, last(v_{\leq k}))$;*
3. *For all nodes $v \in T$ labeled with $\bot$ all children are labeled with $\bot$;*
4. *For every $v \in T$, $a \in \mathrm{Ac}_0$, and $child(T, v)(a) = \{v \cdot x \in child(T, v) \mid x = (a, b)$, for some $b \in \mathrm{Ac}_1\}$, we have that there exists exactly one node in $child(T, v)(a)$ labeled with $\top$ and all the others labeled with $\bot$; i.e., the one labeled with $\top$ corresponds to the action $Player_1$ chooses as a countermove to a chosen by $Player_0$;*
5. *for all $v \in T$, if $|v| > |\mathrm{St}|$ and $V(v) = \top$ then for all $0 \leq i \leq |\mathrm{St}|$ we have that $(\rho)_i \notin W$.*

By the above definition, we formalize the winning condition we consider as follows.

Definition 2.2.10 *Let G be a $2CRG$ and W a set of target states. $Player_0$ (resp., $Player_1$) wins the game G if there is not (resp., there is) a blocking tree for $Player_0$.*

The existence of a blocking tree makes in our setting $Player_0$ losing the game. Conversely, if such a blocking tree does not exist, $Player_0$ wins the game. This condition makes the game

rather than artificial and corresponds to several real settings in formal verification, security, and planning. For example, a scenario in which one wants to check whether a system is immune to an external attack from an adversarial environment can be easily casted in our setting [JM14, JM15, RES+10].

In the rest of the paper we only refer to Definition 2.2.10 as winning condition. However, note that in the sequel we will extend the notion of blocking tree to handle the case of imperfect information along $CRGI$, as we need a richer structure.

2.3 Does the imperfect information matter?

In this section we show, by means of examples, that the imperfect information has a key role to let $Player_0$ winning the game. We use here the definition of blocking tree as given in the previous section and also use an informal explanation of its extension under imperfect information. We prefer to anticipate here this section to help the reader to better understand the formalisms and the constructions we will introduce in the next section as well as to provide some simple reasoning about the game setting we propose.

First, we introduce a $2CRG$ consisting of a variant of the classic *paper, rock, and scissor* game in which $Player_0$ plays as usual, by choosing an action between paper (p), rock (r), and scissor (s), while $Player_1$ uses as actions fire (f) and water (w). The game is depicted in Figure 2.1. The vertexes of the graph are the states of the game and the labels over the edges represent the possible actions that can be taken by the players. The transition function can be easily retrieved by the figure. As this is a one-shot game, we assume that after the first move has been performed, the game remains in the reached state forever, *i.e.* $tr(s_i, a, b) = s_i$ for all $i \in \{1, \cdots, 6\}$, $a \in \mathrm{Ac}_0$, and $b \in \mathrm{Ac}_1$. The set of target states is $\mathrm{W} = \{s_1, s_5, s_6\}$, *i.e.*, the set of states that $Player_0$ wants to reach (note that the target states are drawn in boldface along the figure). One can see that $Player_0$ cannot win the game since there exists a blocking tree for $Player_0$. In fact, by combining the action p for $Player_0$ with the action f for $Player_1$, it prevents the former to reach a target state. The same happens by combining the actions r or s for $Player_0$ with w for $Player_1$. It is important to observe that our game setting is concurrent, *i.e.* in each state every player chooses an action simultaneously and independently from the actions taken by the other players. If we would have considered instead a turn-based scenario, then the order of the players becomes crucial. To better understand this note that in turn-based if $Player_0$ moves first then he loses the game and, conversely, he wins the game if he moves after $Player_1$.

Consider the game as before but with in addition imperfect information on the actions performed by $Player_0$ (see Figure 2.2). Precisely we have that his actions p (*paper*) and r (*rock*) are indistinguishable to $Player_1$, *i.e.* $p \cong r$. Under this assumption we have that $Player_0$ wins the game since there is not a blocking tree for $Player_0$ (built by considering

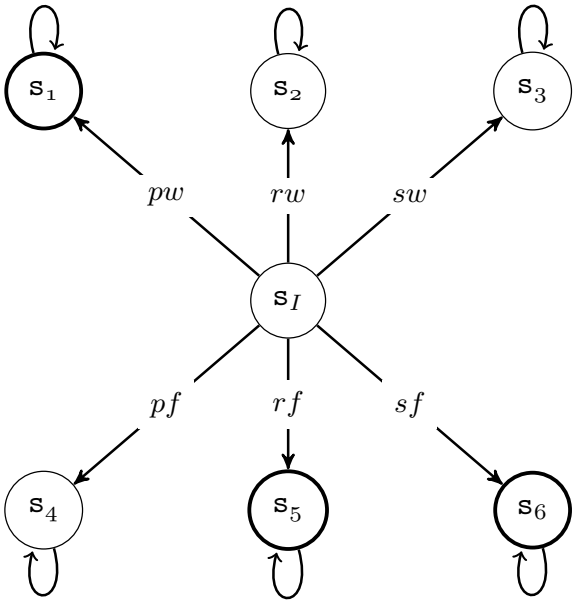


Figure 2.1: Variant of paper, rock, and scissor game.

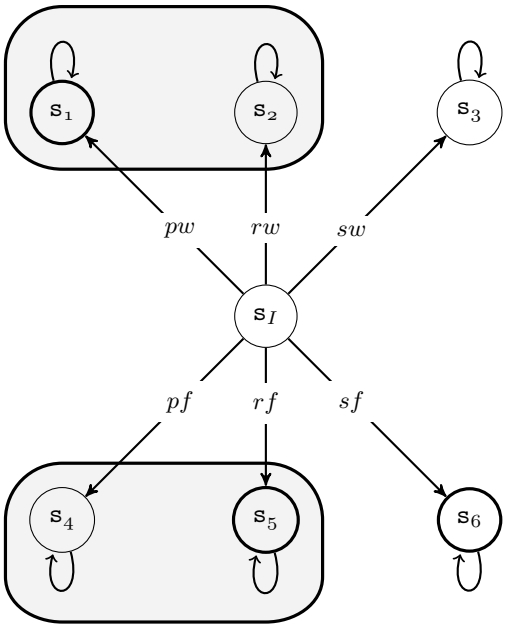


Figure 2.2: Extended paper, rock, and scissor where $p \cong r$.

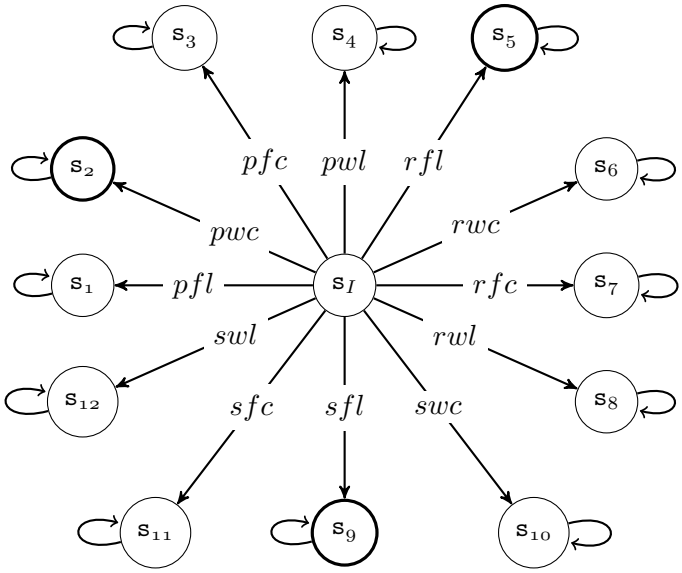


Figure 2.3: Extended paper, rock, and scissor with 3 players.

classic action-choice uniformity for $Player_1$, over the visible actions of $Player_0$). In fact, suppose that $Player_0$ picks an action between p an r. Then, $Player_1$ can use (simultaneously) for both these cases, just one action. If the hidden action is p then he wins the game by choosing f (*fire*). But in case $Player_0$ has chosen r, this would be instead a loosing move for $Player_1$. A similar reasoning applies in case $Player_1$ chooses w (*water*). In other words $Player_1$ cannot uniformly block $Player_0$. So, whereas with perfect information $Player_1$ wins the game, here, having imperfect information regarding some actions, $Player_0$ wins the game.

Finally, as an extension of the above example consider the $CRGI$ depicted in Figure 2.3. In this game we have three players. As above, $Player_0$ can take as action one among p, r, and s, $Player_1$ one between f and w, and additionally $Player_2$ can take actions c for *cloud* or l for *lightning*. We suppose that the moves c and l have the same behavior of w and f, respectively. The transition relation of this game can be easily retrieved by the figure. The set of target states for $Player_0$ is $\mathrm{W} = \{s_2, s_5, s_9\}$. Assume now that $Player_1$ and $Player_2$ have imperfect information on the actions p and r taken by $Player_0$, *i.e.* $p \cong r$. This means that such actions are indistinguishable for $Player_1$ and $Player_2$, making $Player_1$ and $Player_2$ to have only partial view of the game. Over this game, $Player_1$ and $Player_2$ cooperate to win the game. It is not hard to check that by letting $Player_1$ and $Player_2$ to choose actions with different behavior, this makes $Player_0$ to lose the game. With more precise words, one can build a blocking tree by using the described way of acting for $Player_1$ and $Player_2$ and then, in accordance with our definition of winning condition, we have that

$Player_0$ loses the game.

2.4 Automata-Theoretic Solution

In this section, we introduce an automaton-theoretic approach to solve $CRGI$. We start by analyzing the case of 2-player reachability games under imperfect information and show that it is EXPTIME-COMPLETE. Then, we handle the most general setting of $CRGI$ and show that, as for $2CRGI$, deciding the winner of the game is also EXPTIME-COMPLETE. For a matter of clarity, we first recall some basic notation and definitions about automata.

2.4.1 Automata Theory

We recall the definition of *alternating tree automata* and its special case of *nondeterministic tree automaton*.

Definition 2.4.1 *An alternating tree automaton (ATA, for short) is a tuple* $A \triangleq < \Sigma, D, Q, q_0, \delta, F >$*, where* Σ *is the alphabet,* D *is a finite set of directions,* Q *is the set of states,* $q_0 \in Q$ *is the initial state,* $\delta : Q \times \Sigma \rightarrow \mathcal{B}^+(D \times Q)$ *is the transition function, where* $\mathcal{B}^+(D \times Q)$ *is the set of all positive Boolean combinations of pairs* (d, q) *with* d *direction and* q *state, and* $F \subseteq Q$ *is the set of the accepting states.*

An ATA A recognizes (finite) trees by means of runs. For a Σ-labeled tree $< T, V >$, with $T = D^*$, a run is a $(D^* \times Q)$-labeled $\mathbb{N}$-tree $< T_r, r >$ such that the root is labeled with (ε, q_0) and the labels of each node and its successors satisfy the transition relation.

For example, assume that A, being in a state q, is reading a node x of the input tree labeled by ξ. Assume also that $\delta(q, \xi) = ((0, q_1) \vee (1, q_2)) \wedge (1, q_1)$. Then, there are two ways along which the construction of the run can proceed. In the first option, one copy of the automaton proceeds in direction 0 to state q_1 and one copy proceeds in direction 1 to state q_1. In the second option, two copies of A proceed in direction 1, one to state q_1 and the other to state q_2. Hence, $\vee$ and $\wedge$ in $\delta(q, \xi)$ represent, respectively, choice and concurrency. A run is *accepting* if all its leaves are labeled with accepting states. An input tree is accepted if there exists a corresponding accepting run. By $L(A)$ we denote the set of trees accepted by A. We say that A is not empty if $L(A) \neq \emptyset$.

As a special case of alternating tree automata, we consider *nondeterministic tree automata* (NTA, for short), where the concurrency feature is not allowed. That is, whenever the automaton visits a node x of the input tree, it sends to each successor (direction) of x at most one copy of itself. More formally, an NTA is an ATA in which δ is in disjunctive normal form, and in each conjunctive clause every direction appears at most once.

2.4.2 Solution for 2CRGI

Recall that blocking trees are projections of decision trees. In case of imperfect information over the actions played by $Player_0$, some non-uniform strategies of $Player_1$ are no longer valid. This reflects directly in the way the blocking tree is built. To restrict to the visibility of the players, we merge in the blocking tree the directions that come out from indistinguishable actions. This means that the tree branching is reduced accordingly.

In other words, in the perfect information setting, the blocking tree has just one direction for each possible action of $Player_0$. In the imperfect information setting, instead, we consider a "thin" tree in which some nodes carry more action choices (all indistinguishable) at the same time. So, the set of directions of the blocking tree is given by $\cong$ and set to $[\mathrm{Ac}_0]$. In the 2-player case, the set $[\mathrm{Ac}_0]$ represents the class of actions Ac_0 that are indistinguishable to $Player_1$.

For the solution side, we use an automata-approach via alternating tree automata. The idea is to read a $\{\top, \bot\}$-labeled full $([\mathrm{Ac}_0] \times \mathrm{Ac}_1)$*-tree such that more copies of the automaton are sent to the same directions along the class of equivalence over $[\mathrm{Ac}_0]$. The automaton checks the consistency of the moves on the fly and its size is just polynomial in the size of the game arena. These trees are taken with depth greater than the number of states; so if no state in W is reached in $|\mathrm{St}|$ step, then there is a loop over the states in the game model that forbids to reach states in W in the future.

Theorem 2.4.1 *Given a $2CRGI$ G played by $Player_0$ and $Player_1$, the problem of deciding whether $Player_0$ wins the game is* EXPTIME-COMPLETE.

Proof sketch. Let G be a $2CRGI$. For the lower bound, we recall that deciding the winner in a 2-player turn-based games with imperfect information is EXPTIME-HARD [Rei84]. For the upper bound, we use an automata-theoretic approach. Precisely, we build an ATA A that accepts all trees that are blocking for $Player_0$ over G. These are $\{\top, \bot\}$-labeled $([\mathrm{Ac}_0] \times \mathrm{Ac}_1)^*$-trees that represent the projection of the decision tree over the blocking tree in accordance with the visibility over the actions. The branching degree of the input tree is thus given by $[\mathrm{Ac}_0] \times \mathrm{Ac}_1$. The automaton, therefore will send more copies on the same direction of the input tree when they correspond to hidden actions. Then it will check the consistency with the states on the fly by taking in consideration the information stored in the node of the tree. The automaton accepts only trees that have depth (*i.e.* all its paths) greater than $|\mathrm{St}|$. This can be simply checked by means of a binary counter along with the states of the automaton. For the sake of readability we omit this part.

The automaton uses as set of states $Q = \mathrm{St} \times \mathrm{St} \times \{\top, \bot\} \times \{0, 1\}$ and alphabet $\Sigma = \{\top, \bot\}$. We use in Q a duplication of game states as we want to remember the game state associated to the parent node while traversing the tree. For the initial state we set $q_0 = (s_I, s_I, \top, 0)$, *i.e.* for simplicity the parent game state associated to the root of the tree

is the game state itself. The flag $f \in \{0, 1\}$ indicates whether along a path we have entered a target state. In that case we move f from 0 to 1. Given a state $s = (p, q, t, f)$ and symbol t', the transition relation $\delta(s, t')$ is defined as:

$$\begin{cases} \bigwedge_{a_0 \in \mathrm{Ac}_0} \bigwedge_{a_1 \in \mathrm{Ac}_1} (d, (q, q', \top, f')) & \text{if } t = t' = \top \wedge f = 0; \\ \text{true} & \text{if } t' = \bot; \\ \text{false} & \text{if } t = \bot \vee f = 1. \end{cases}$$

where $q' = tr(q, a_0, a_1)$, $t, t' \in \{\top, \bot\}$, $d = [\mathrm{Ac}_0] \times \mathrm{Ac}_1$, and $f' = 1$ if $q' \in \mathrm{W}$ otherwise $f' = f$.

The set of accepted states is $\mathrm{F} = \{(p, q, t, f) : p, q \in \mathrm{St} \wedge t = \top \wedge f = 0\}$. Recall that an input tree is accepted if there exists a run whose leaves are all labeled with accepting states. In our setting this means that an input tree simulates a blocking tree for $Player_0$. So, if the automaton is empty then $Player_0$ wins the game, *i.e.*, does not exist a blocking tree for him. The required computational complexity of the solution follows by considering that: *(i)* the size of the automaton is polynomial in the size of the game, *(ii)* to check its emptiness can be performed in exponential time over the number of states [EJ88, KVW00].□

2.4.3 Solution for CRGI

In this section we describe the main result of this work, *i.e.* an exponential solution algorithm to decide $CRGI$s. As we have anticipated earlier we use an opportune extension of the automata-theoretic approach we have introduced in the previous sections. Such an extension needs to work with n players: $Player_0 \ldots Player_{n-1}$.

In particular, we decide the game by looking for a blocking tree for $Player_0$, which is built by considering all possible ways players $Player_j$, with $1 \leq j < n$, have to block $Player_0$, but playing under uniform visibility. These trees, once again, can be collected in a ATA that we can build by opportunely extending the one introduced for 2-player games with imperfect information. Precisely, the automaton will take as input $\{\top, \bot\}$-labeled full $([\mathrm{Ac}_0] \times \mathrm{Ac}_1 \times \ldots \times \mathrm{Ac}_{n-1})^*$-trees such that more copies of the automaton are sent along the same directions as defined by the equivalence class over the actions.

Theorem 2.4.2 *Given a CRGI G played by $Player_0 \ldots Player_{n-1}$, the problem of deciding whether $Player_0$ wins the game is* EXPTIME-COMPLETE.

Proof sketch. Let G be a $CRGI$. For the lower bound, we inherit it from $2CRGI$. For the upper bound, we build an ATA A that accepts all trees that are blocking for $Player_0$ over G. These are $\{\top, \bot\}$-labeled full $([\mathrm{Ac}_0] \times \mathrm{Ac}_1 \times \ldots \times \mathrm{Ac}_{n-1})^*$-trees that represent all projections, one for each $Player_i$, in accordance with the visibility of the actions. The branching degree of the input tree is thus given by $[\mathrm{Ac}_0] \times \mathrm{Ac}_1 \times \ldots \times \mathrm{Ac}_{n-1}$. The automaton, as in the 2-player

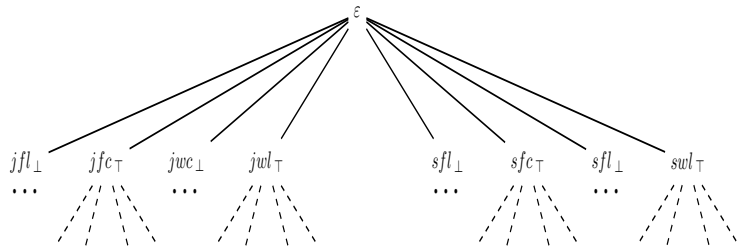


Figure 2.4: Tree accepted by the automaton.

case, has set of states $Q = \mathrm{St} \times \mathrm{St} \times \{\top, \bot\} \times \{0, 1\}$, initial state $q_0 = (s_I, s_I, \top, 0)$, and alphabet $\Sigma = \{\top, \bot\}$. Given a state $s = (p, q, t, f)$ and symbol t', the transition relation $\delta(s, t')$ is defined as:

$$\begin{cases} \bigwedge_{a_0 \in \mathrm{Ac}_0} \cdots \bigwedge_{a_{n-1} \in \mathrm{Ac}_{n-1}} (d, (q, q', \top, f')) & \text{if } t = t' = \top \wedge f = 0; \\ \text{true} & \text{if } t' = \bot; \\ \text{false} & \text{if } t = \bot \vee f = 1. \end{cases}$$

where $q' = tr(q, a_0, \ldots, a_{n-1})$, $t, t' \in \{\top, \bot\}$, $d = [\mathrm{Ac}_0] \times \ldots \times \mathrm{Ac}_{n-1}$, and $f' = 1$ if $q' \in \mathrm{W}$ otherwise $f' = f$.

The set of accepted states as for the 2-player case is $\mathrm{F} = \{(p, q, t, f) : p, q \in \mathrm{St} \wedge t = \top \wedge f = 0\}$. By applying a reasoning similar to that used in the previous section, one can see that the automaton A accepts only trees that simulate blocking trees for $Player_0$. So, if the automaton is empty then $Player_0$ wins the game. We finally obtain the required complexity result by observing that also in this case the size of the automaton is polynomial and by recalling that checking its emptiness can be done in exponential time over the number of states [EJ88, KVW00]. So, on the construction of the automata, any branching degree, even exponential, is not a problem.□

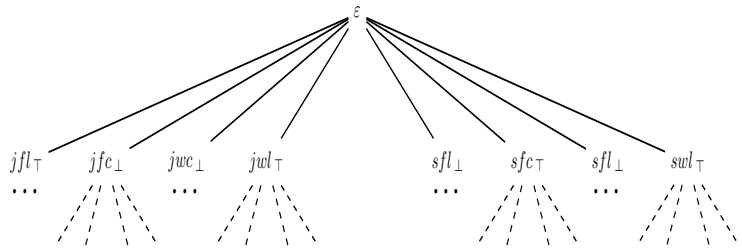


Figure 2.5: Tree rejected by the automaton.

We conclude this section by reasoning on the application of the above automata construction over the game example reported in Figure 2.3. First observe that, $[\mathrm{Ac}_0] = \{j, s\}$ (j is the representative action of $p \cong r$). One can see that the automaton accepts the $\{\top, \bot\}$-labeled full $([\mathrm{Ac}_0] \times \mathrm{Ac}_1 \times \mathrm{Ac}_2)^*$-tree depicted in Figure 2.4 but rejects the one in Figure 2.5. Indeed, the projection of the tree in Figure 2.4 over the decision tree of the game induces a blocking tree for $Player_0$ in which all paths do not reach a target state. Conversely, the projection of the tree in Figure 2.5 over the decision tree of the game induces a tree in which there exists a path (precisely the path leading to rfl) that reaches a target state.

2.5 Conclusion

On game reasoning for multi-player systems, imperfect information plays a key role. Several fundamental works in formal verification and strategy reasoning have deeply investigated this setting. Among the others we mention the seminal work of Pnueli and Rosner [PR90] that considered n-player games, by extending important results achieved by Reif [Rei84] over two-player games under imperfect information. Pnueli and Rosner considered multi-player games over different architecture models. Worth of note is the pipeline of processes in which each output communication of a process i is used as an input to a process $i + 1$. Another seminal work concerns ATL by Alur, Kupferman and Henzinger [AHK02], who addressed the imperfect information problem from a logic point of view. This setting has been an important source of several works in AI and formal verification.

However, moving from perfect to imperfect information makes the problem of deciding multi-agent games much more complicated. For example, the reachability game under the pipeline architecture of Pnueli and Rosner is non-elementary and solving ATL goals specifications over multi-agent concurrent game structures is undecidable [AHK02, DT11] (in the general setting). In the perfect information case, instead, they are both elementary decidable. This has given rise in the years to the need of investigating restricted imperfect information settings (and thus methodologies) in which the decision problem gets back to an elementary complexity and, possibly, not too far from the one for the perfect information case.

In this chapter we have addressed a variant of the reachability game problem for n players under a specific form of imperfect information. Precisely we have considered the case in which $Player_0$ is omniscient and plays against all other players who have common partial visibility over the actions he can perform. Remarkably, we have considered as a winning condition for $Player_0$ the inability for all other players to prevent him to reach a target state (while using uniformity along action choice) and formalized this concept by introducing *blocking trees*. As a variant of classic reachability condition in 2-player concurrent games, this enforces $Player_0$ ability to win the game and so to declare always a winner of the game.

We have proved that our game setting can be decided in EXPTIME by making use of an automata-theoretic solution. It is worth remarking the efficiency of the automata solution we have provided that is able to handle several memoryfull player's strategies under imperfect information all in one shot.

Overall, the framework we have addressed is one of the few multi-player game settings with imperfect information yet elementary decidable. It is important to note that, one cannot translate our game with n players in a two-player one, by just removing players. In particular this is not possible for the imperfect information. To be convinced, see the very end of last paragraph of Section 2.3: by simply merging $Player_1$ and $Player_2$ (performing only one action at time), then $Player_0$ loses the game.

We argue that the introduced game framework has several practical and broad applications. Along the introduction we have given some specific example. In addition, one can think of a rob and copper scenario in which several independent and non-communicating coppers try to catch a robber being at different distances from him. Clearly the coppers in the back have less information from the ones being in the front and the robber, being in front of every one else and playing adversarial, can have full information over the actions of the other players. In such a scenario, a reasonable goal for the coppers is to prevent the robber to reach a safe (target) state.

Another way to see our work is an orthogonal application of the module checking extension along with multiple-agents [JM14, JM15]. By casting that settings in ours, we address the case in which the system is represented by $Player_0$ and the environment is made by several agents (the opponent players) having imperfect information about the system. We recall that in [JM14, JM15] the environment is modeled by a single player while the system is composed by several agents.

Clearly, there are several other specific settings/extensions one can consider for n players under imperfect information. We conclude this section just reporting some of them, which we aim to investigate as future work. One extension that would be worth investigating concerns the relaxation of the common visibility among the opponent players upon $Player_0$. Another interesting extension concerns multi-target games. That is every player has its own target to reach. In this case every player works against every one else. To give some fairness condition over the game, one can also think of having an order (for each player) over the targets. This means that if a player cannot reach his own goal, he may want to help one player rather than another. This can be generalized by considering a solution concept as a target. We conjecture that the exponential algorithm we have proposed can be adapted to deal with this scenario as well.

Part II

Counting Strategies

CHAPTER 3

Reasoning about Graded Strategy Quantifiers

Contents

3.1 Introduction

Formal methods in system design are a renowned story of success. Breakthrough contributions in this field comprise *model checking* [CE81, QS82] and *temporal logics* such as LTL [Pnu77], CTL [CE81], CTL* [EH86], and the like. First applications of these methodologies involved *closed systems* [HP85] generally analyzing whether a Kripke structure, modeling the system, meets a temporal logic formula, specifying the desired behavior [CGP02]. In the years several algorithms have been proposed in this setting and some implemented as tools [BBF$^+$10]. Nevertheless these approaches turn to be useless when applied to *open systems* [HP85]. The latter are characterized, in the simplest situation, by an ongoing interaction with an external environment on which the whole system behavior deeply relies. To be able to deal with the unpredictability of the environment, extensions of the basic verification techniques have come out. A first attempt worth of note is *module checking* where a Kripke structure is replaced by a specific two-player arena. Module checking has been first introduced in [KV96, KVW01]. In the last decade this methodology has been fruitfully extended in several directions (see [ALM$^+$13, JM14, JM15] for some related works).

Starting from the study of module checking, researchers have looked for logics focusing on the *strategic behavior* of players in *multi-agent systems* [AHK02]. One of the most important developments in this field is *Alternating-Time Temporal Logic* (ATL*, for short), introduced by Alur, Henzinger, and Kupferman [AHK02]. This logic allows to reason about strategies of agents having the satisfaction of temporal goals as the payoff criterion. Formally, it is obtained as a generalization of CTL*, in which the existential `E` and the universal `A` *path quantifiers* are replaced with *strategic modalities* of the form $\langle\langle \mathrm{A} \rangle\rangle$ and $[\![\mathrm{A}]\!]$, where A is a set of *agents*. Strategic modalities over agent teams are used to describe cooperation and competition among them in order to achieve certain goals. In particular, these modalities express selective quantifications over those paths that are the result of infinite interaction between a coalition and its complement.

Despite its expressiveness, ATL* suffers from the strong limitation that strategies are treated only implicitly in the semantics of its modalities. This restriction makes the logic less suited to formalize several important solution concepts, such as *Nash Equilibrium*. These considerations led to the introduction of *Strategy Logic* (SL, for short) [CHP07, MMV10a], a more powerful formalism for strategic reasoning. As a key aspect, SL treats strategies as *first-order objects* that can be determined by means of the existential $\langle\langle x \rangle\rangle$ and universal $[\![x]\!]$ quantifiers, which can be respectively read as *"there exists a strategy x"* and *"for all strategies x"*. Remarkably, a strategy in SL is a generic conditional plan that at each step prescribes an action on the base of the history of the play. Such a plan is not intrinsically glued to a specific agent but an explicit binding operator (a, x) allows to link an agent a to the strategy associated with a variable x.

A common aspect about all logics mentioned above is that quantifications are either existential or universal. *Per contra*, there are several real scenarios in which "more precise" quantifications are crucially needed (see [BMM12, MMS15], for an argumentation). This has attracted the interest of the formal verification community to *graded modalities*. These have been first studied in classic modal logic [Fin72] and then exported to the field of *knowledge representation* to allow quantitative bounds on the set of individuals satisfying specific properties. Specifically, they are *counting quantifiers* in first-order logics [GOR97], *number restrictions* in *description logics* [LST05, LWW07, CDL99, ABL07] and *numerical constraints* in query languages [Bár15].

First applications of graded modalities in formal verification concern closed systems. In [KSV02], *graded* μCALCULUS has been introduced in order to express statements about a given number of immediately accessible worlds. Successively in [FNP09b, BMM09, BMM10, BMM12], the notion of graded modalities have been extended to deal with number of paths. Among the others graded CTL (GCTL, for short) has been introduced with a suitable axiomatization of counting [BMM12]. This work has been recently extended in [AMR15] to address GCTL*, a graded extension of CTL*.

In open systems verification, we are aware of just two orthogonal approaches in which graded modalities have been investigated, but in a very restricted form: module checking for graded μCALCULUS [FMP08] and an extension of ATL with graded path modalities (GATL, for short) [FNP09a]. In particular, the former involves a counting of one-step moves among two agents, the latter allows for a more restricted counting on the histories of the game, but in a multi-player setting. Both approaches suffer of several limitations. First, not surprisingly, they cannot express powerful game reasoning due to the limitation of the underlying logic. Second, it is based on a very rigid and restricted counting of strategies.

In this chapter, we take a different approach by formally introducing a machinery to count strategies in a multi-agent setting and use it upon the powerful framework of SL. Precisely, we introduce and study *Graded Strategy Logic* (GSL) which extends SL with the existential $\langle\!\langle x \geq g \rangle\!\rangle \varphi$ and universal $[\![x < g]\!]\varphi$ graded strategy quantifiers. They allow to express that there are *at least* g or *all but less than* g strategies x satisfying φ, respectively. As in SL, we use the binding operator to associate these strategies to agents.

As far as the counting of strategies is concerned, one of the main difficulties resides on the fact that some strategies, although looking different, produce the same outcome and therefore have to be counted as one. To overcome this problem while preserving a correct counting over paths for the underlining logic SL, we introduce a suitable equivalence relation over profiles based on the strategic behavior they induce. This is by its own an important contribution of this work.

To show the applicability of GSL we investigate basic game-theoretic and verification questions over a powerful fragment of GSL. Recall that model checking is non-elementary-

complete for SL and this has spurred researchers to investigate fragments of the logic for practical applications. Here, we concentrate on the *vanilla* version of the SL[1G] fragment of SL. We recall that SL[1G] was introduced in [FAGV12]. As for ATL, vanilla SL[1G] (for the first time introduced here) requires that two successive temporal operators in a formula are always interleaved by a strategy quantifier. We prove that the model-checking problem for this logic is PTIME-COMPLETE. We also show positive results about the determinacy of turn-based games.

GSL can have useful applications in several multi-agent game scenarios. For example, in safety-critical systems, it may be worth knowing whether a controller agent has a redundant winning strategy to play in case of some fault. Having more than a strategy may increase the chances for a success [ATO+09], *i.e.*, if a strategy fails for any reason, it is possible to apply the others.

Such a redundancy can easily be expressed in GSL by requiring that at least two different strategies exist for the achievement of the safety goal. The universal graded strategy quantifier may turn useful to grade the "security" of a system. For example, one can check whether preventing the use of at most k strategies, the remaining ones are all winning. In a network this may correspond to prevent some attacks while leaving the communication open.

3.2 Graded Strategy Logic

In this section we introduce syntax and semantics of *Graded Strategy Logic* (GSL, for short), an extension of *Strategy Logic* (SL, for short) [MMV10a] that allows reasoning about the number of strategies an agent may exploit in order to satisfy a given temporal goal. We recall that SL simply extends LTL with two strategy quantifiers and a binding construct used to associate an agent to a strategy.

This section is organized as follows. In Subsection 3.2.1, we recall the definition of concurrent game structure, used to interpret GSL and give some examples. In Subsection 3.2.2 we introduce the syntax of GSL, and, in Subsection 3.2.3, its semantics. Finally, in Subsection 3.2.4 we list the main results of this work.

3.2.1 Model

Similarly to SL, as semantic framework we use *concurrent game structures* [AHK02], *i.e.*, a generalization of both *Kripke structures* [Kri63] and *labeled transition systems* [Kel76] in which the system is modeled as a game where players perform actions chosen strategically as a function on the history of the play.

Definition 3.2.1 (Concurrent Game Structure) *A* Concurrent Game Structure *(*CGS*, for short) is a tuple* $\mathcal{G} \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_{\mathrm{o}} \rangle$*, where* AP*,* Ag*,* Ac*, and* St *are sets of*

atomic propositions, agents/players, actions *and* states, *respectively,* $s_I \in \mathrm{St}$ *is an* initial state, *and* $\mathsf{ap} : \mathrm{St} \to 2^{\mathrm{AP}}$ *is a* labeling function *mapping each state to the set of atomic propositions true in that state. Let* $\mathrm{Dc} \triangleq \mathrm{Ag} \rightharpoonup \mathrm{Ac}$ *be the set of* decisions, i.e., *partial functions describing the choices of an action by some agent. Then,* $\mathsf{tr} : \mathrm{Dc} \to (\mathrm{St} \rightharpoonup \mathrm{St})$ *denotes the* transition function *mapping every decision* $\delta \in \mathrm{Dc}$ *to a partial function* $\mathsf{tr}(\delta) \subseteq \mathrm{St} \times \mathrm{St}$ *representing a deterministic graph over the states.*

Intuitively, a CGS can be seen as a generic *labeled transition graph* [Kel76], where labels are possibly incomplete agent decisions, which determine the transitions to be executed at each step of a play in dependence of the choices made by the agents in the relative state. In particular, incomplete decisions allow us to represent any kind of legal move in a state, where some agents or a particular combination of actions may not be active. It is worth noting that, due to the way the transition function is defined, a CGS is in general nondeterministic. Indeed, two different but indistinguishable decisions may enable different transitions for the same state. Even more, a single decision may induce a non-functional relation. However, due to the focus of this work, we restrict to the case of deterministic games structures, by describing later on few conditions that rule out how the transition function has to map partial decisions to transitions.

A concurrent game structure $\mathcal{G}$ naturally induces a graph $\mathsf{Gr}(\mathcal{G}) = \langle \mathrm{St}, Ed \rangle$, whose vertexes are represented by the states and the edge relation $Ed \triangleq \bigcup_{\delta \in \mathrm{Dc}} \mathsf{tr}(\delta)$ is obtained by rubbing out all labels on the transitions. Note that there could be states where no transitions are available, *i.e.*, $\mathsf{dom}(Ed) \subset \mathrm{St}$. If this is the case, those states in $\mathrm{St} \setminus \mathsf{dom}(Ed)$ are called *sink-states*. A *path* $\pi \in \mathrm{Pth} \triangleq \{\pi \in \mathrm{St}^{\omega} : \forall i \in \mathbb{N} \,.\, ((\pi)_i, (\pi)_{i+1}) \in Ed\}$ is simply an infinite path in $\mathcal{G}$. Similarly, the *order* $|\mathcal{G}| \triangleq |\mathsf{Gr}(\mathcal{G})|$ (resp., *size* $\|\mathcal{G}\| \triangleq \|\mathsf{Gr}(\mathcal{G})\|$) of $\mathcal{G}$ is the order (resp., size) of its induced graph. As usual in the study of extensive-form games, finite paths also describe the possible evolutions of a play up to a certain point. For this reason, they are called in the game-theoretic jargon *histories*, whose corresponding set is denoted by $\mathrm{Hst} \triangleq \{\rho \in \mathrm{St}^* : \forall i \in [0, |\rho| - 1[\,.\, ((\rho)_i, (\rho)_{i+1}) \in Ed\}$. Moreover, by $\mathsf{fst}(\rho) = \rho_0$ (resp., $\mathsf{fst}(\pi) = \pi_0$) we denote the first element in the history (resp., path), by $\mathsf{lst}(\rho)$ we denote the last element occurring in the history ρ and by $\rho_{\leq i}$ (resp., $\pi_{\leq i}$) we denote the prefix up to the state of index i. We now introduce the sets of decisions, agents, and actions that trigger some transition in a given state $s \in \mathrm{St}$ by means of the three functions $\mathsf{dc} : \mathrm{St} \to 2^{\mathrm{Dc}}$, $\mathsf{ag} : \mathrm{St} \to 2^{\mathrm{Ag}}$, and $\mathsf{ac} : \mathrm{St} \times \mathrm{Ag} \to 2^{\mathrm{Ac}}$ such that:

$$\mathsf{dc}(s) \triangleq \{\delta \in \mathrm{Dc} : s \in \mathsf{dom}(\mathsf{tr}(\delta))\};$$

$$\mathsf{ag}(s) \triangleq \{a \in \mathrm{Ag} : \exists \delta \in \mathsf{dc}(s) \,.\, a \in \mathsf{dom}(\delta)\};$$

$$\mathsf{ac}(s, a) \triangleq \{\delta(a) \in \mathrm{Ac} : \delta \in \mathsf{dc}(s) \ \wedge \ a \in \mathsf{dom}(\delta)\}, \text{ for all } a \in \mathrm{Ag}.$$

These functions can be easily lifted to the set of histories as follows: $\mathsf{dc} : \mathrm{Hst} \to 2^{\mathrm{Dc}}$ with $\mathsf{dc}(\rho) \triangleq \mathsf{dc}(\mathsf{lst}(\rho))$, $\mathsf{ag} : \mathrm{Hst} \to 2^{\mathrm{Ag}}$ with $\mathsf{ag}(\rho) \triangleq \mathsf{ag}(\mathsf{lst}(\rho))$, and $\mathsf{ac} : \mathrm{Hst} \times \mathrm{Ag} \to 2^{\mathrm{Ac}}$ with $\mathsf{ac}(\rho, a) \triangleq \mathsf{dc}(\mathsf{lst}(\rho), a)$.

A decision $\delta \in \mathrm{Dc}$ is *coherent w.r.t.* a state $s \in \mathrm{St}$ (*s-coherent*, for short), if $\mathsf{ag}(s) \subseteq \mathsf{dom}(\delta)$ and $\delta(a) \in \mathsf{ac}(s, a)$, for all $a \in \mathsf{ag}(s)$. By $\mathrm{Dc}(s) \subseteq \mathrm{Dc}$, we denote the set of all s-coherent decisions.

A *strategy* is a partial function $\sigma \in \mathrm{Str} \triangleq \mathrm{Hst} \rightharpoonup \mathrm{Ac}$ prescribing, whenever defined, which action has to be performed for a certain history of the current outcome. Roughly speaking, it is a generic conditional plan which specifies *"what to do"* but not *"who will do it"*. Indeed, a given strategy can be used by more than one agent at the same time. We say that σ is *coherent w.r.t.* an agent $a \in \mathrm{Ag}$ (*a-coherent*, for short) if, in each possible evolution of the game, either a is not influential or the action that σ prescribes is available to a. Formally, for each history $\rho \in \mathrm{Hst}$, it holds that either $a \notin \mathsf{ag}(\rho)$ or $\rho \in \mathsf{dom}(\sigma)$ and $\sigma(\rho) \in \mathsf{ac}(\rho, a)$. By $\mathrm{Str}(a) \subseteq \mathrm{Str}$ we denote the set of a-coherent strategies. Moreover, $\mathrm{Str}(\mathrm{A}) \triangleq \bigcap_{a \in \mathrm{A}} \mathrm{Str}(a)$ indicates the set of strategies that are coherent with all agents in $\mathrm{A} \subseteq \mathrm{Ag}$.

For a state $s \in \mathrm{St}$, we say that σ is *s-total* iff it is defined on all non-trivial histories (*i.e.*, $|\rho| > 0$) starting in s, *i.e.*,$\mathsf{dom}(\sigma) = \{\rho \in \mathrm{Hst} | \mathsf{fst}(\rho) = s\}$.

A *profile* is a function $\xi \in \mathrm{Prf} \triangleq \mathrm{Ag} \to_a \mathrm{Str}(a)$ specifying a unique behavior for each agent $a \in \mathrm{Ag}$ by associating it with an *a-coherent* strategy $\xi(a) \in \mathrm{Str}(a)$. Given a profile ξ, to identify which action an agent $a \in \mathrm{Ag}$ has chosen to perform on a history $\rho \in \mathrm{Hst}$, we first extract the corresponding strategy $\xi(a)$ and then determine the action $\xi(a)(\rho)$, whenever defined. To identify, instead, the whole decision on ρ, we apply the standard flipping operator to ξ. [1] We get so a function $\widehat{\xi} : \mathrm{Hst} \to \mathrm{Dc}$ such that $\widehat{\xi}(\rho)(a) = \xi(a)(\rho)$, which maps each history to the planned decision.

A path $\pi \in \mathrm{Pth}$ is a *play w.r.t.* a profile $\xi \in \mathrm{Prf}$ (*ξ-play*, for short) iff, for all $i \in [0, |\pi|[$, there exists a decision $\delta \in \mathsf{dc}((\pi)_i)$ such that $\delta \subseteq \widehat{\xi}((\pi)_{\leq i})$ and $((\pi)_i, (\pi)_{i+1}) \in \mathsf{tr}(\delta)$, *i.e.* $(\pi)_{i+1}$ is one of the successors of $(\pi)_i$ induced by the decision $\widehat{\xi}((\pi)_{\leq i})$ prescribed by the profile ξ on the history $(\pi)_{\leq i}$.

CGSs describe generic mathematical structures, where the basilar game-theoretic notions of history, strategy, profile, and play can be defined. However, in several contexts, some constraints rule out how the function tr maps partial decisions to transitions between states. Here, as already observed, we require that the CGSs are deterministic. We do this by means of the following constraints:

1. there are no sink-states, *i.e.*, $\mathsf{dc}(s) \neq \varnothing$, for all $s \in \mathrm{St}$;

2. for all s-coherent decisions $\delta \in \mathrm{Dc}(s)$, there exists a set of agents $\mathrm{A} \subseteq \mathsf{ag}(s)$ such that $\delta_{\restriction \mathrm{A}} \in \mathsf{dc}(s)$;

[1] By $\widehat{g} : (\mathrm{B} \to (\mathrm{A} \to \mathrm{C}))$ we denote the operation of flipping of a function $g : (\mathrm{A} \to (\mathrm{B} \to \mathrm{C}))$.

3. each decision induces a partial function among states, *i.e.* $\mathsf{tr}(\delta) \in \mathrm{St} \rightharpoonup \mathrm{St}$, for all $\delta \in \mathrm{Dc}$;

4. there are no different but indistinguishable active decisions in a given state $s \in \mathrm{St}$, *i.e.*, for all $\delta_1, \delta_2 \in \mathsf{dc}(s)$ with $\delta_1 \neq \delta_2$, there exist $a \in \mathsf{dom}(\delta_1) \cap \mathsf{dom}(\delta_2)$ such that $\delta_1(a) \neq \delta_2(a)$.

Given a state $s \in \mathrm{St}$, the determinism in a CGS ensures that there exists exactly one ξ-play π starting in s, *i.e.*, $\mathsf{fst}(\pi) = s$. Such a play is called (ξ, s)-*play*. For this reason, we use the *play function* $\mathsf{play} : \mathrm{Prf} \times \mathrm{St} \to \mathrm{Pth}$ to identify, for each profile $\xi \in \mathrm{Prf}$ and state $s \in \mathrm{St}$, the corresponding (ξ, s)-*play* $\mathsf{play}(\xi, s)$.

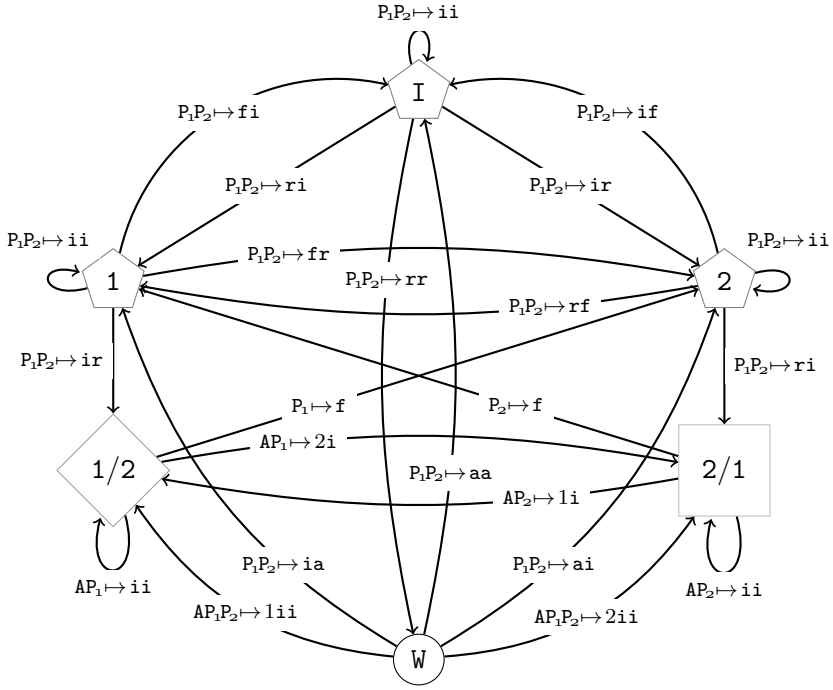


Figure 3.1: A scheduler system $\mathcal{G}_S$.

As a running example, consider the concurrent game structure $\mathcal{G}_S$ depicted in Figure 3.1. It models a *scheduler system* that comprises two *processes*, $\mathtt{P}_1$ and $\mathtt{P}_2$, willing to access a *shared resource*, such as a processor, and an *arbiter* $\mathtt{A}$ used to solve conflicts arisen under contending requests. The processes can use four actions: `i` for idle, `r` for (resource) request, `f` for free (a resource), and `a` for abandon (a pending request), all with the obvious meaning. The action `i` means that the process does not want to change the current situation in which the entire system resides. The action `r` is used to ask for the resource, when this is not yet owned, while the action `f` releases it. Finally, the action `a` is asserted by a process that, although has asked for the resource, did not obtain it and so it decides to relinquish the request. The system can reside in the states `I`, `1`, `2`, `1/2`, `2/1` and `W`. The first three are ruled by the processes, the

last by all the agents, and $\mathtt{1/2}$ (resp, $\mathtt{2/1}$) by $\mathtt{P_1}$ (resp., $\mathtt{P_2}$) and $\mathtt{A}$. The idle state $\mathtt{I}$ indicates that none of the processes owns the resource, while a state $k \in \{\mathtt{1}, \mathtt{2}\}$ asserts that process $\mathtt{P}_k$ is using it. The state $\mathtt{1/2}$ (resp. $\mathtt{2/1}$) indicates that the process $\mathtt{P_1}$ (resp., $\mathtt{P_2}$) has the resource, while its competitor requires it. Finally, the waiting state $\mathtt{W}$ represents the case in which an action from the arbiter is required in order to solve a conflict. To denote who is the owner of the resource, we label $\mathtt{1}$ and $\mathtt{1/2}$ (resp., $\mathtt{2}$ and $\mathtt{2/1}$) with the atomic proposition $\mathtt{r_1}$ (resp., $\mathtt{r_2}$). A decision is graphically represented by $\vec{\mathtt{a}} \mapsto \vec{\mathtt{c}}$, where $\vec{\mathtt{a}}$ is a sequence of agents and $\vec{\mathtt{c}}$ is a sequence of corresponding actions. For example $\mathtt{P_1P_2} \to \mathtt{ir}$ indicates that agents $\mathtt{P_1}$ and $\mathtt{P_2}$ take actions $\mathtt{i}$ and $\mathtt{r}$, respectively. All the other available decisions are depicted in Figure 3.1.

3.2.2 Syntax

GSL extends SL by replacing the two classic *strategy quantifiers* $\langle\!\langle x \rangle\!\rangle$ and $[\![x]\!]$, where x belongs to a countable set Vr of variables, with their graded version $\langle\!\langle x \geq g \rangle\!\rangle$ and $[\![x < g]\!]$, where the finite number $g \in \mathbb{N}$ denotes the corresponding *degree*, that is a bound associated to the strategy quantifiers. Intuitively, these quantifiers are read as *"there exist at least g strategies"* and *"all but less than g strategies"*. Moreover, GSL syntax comprises a set AP of atomic proposition to expresses properties over the states, a binding operator to link strategies to agents, and Boolean connectives.

Definition 3.2.2 (GSL **Syntax)** GSL formulas *are built inductively by means of the following context-free grammar, where $a \in \mathrm{Ag}$, $p \in \mathrm{AP}$, $x \in \mathrm{Vr}$, and $g \in \mathbb{N}$:*

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathsf{R} \varphi \mid \langle\!\langle x \geq g \rangle\!\rangle \varphi \mid [\![x < g]\!]\varphi \mid (a, x)\varphi.$$

As usual, to provide the semantics of a predicative logic, it is necessary to define the concept of free and bound *placeholders* of a formula. As for SL, since strategies can be associated to both agents and variables, we need the set of *free agents/variables* $\mathsf{free}(\varphi)$ as the subset of $\mathrm{Ag} \cup \mathrm{Vr}$ containing *(i)* all agents a for which there is no binding (a, x) before the occurrence of an atomic proposition and *(ii)* all variables x for which there is a binding (a, x) but no quantification $\langle\!\langle x \geq g \rangle\!\rangle$ or $[\![x < g]\!]$.

Definition 3.2.3 (GSL **Free Agents/Variables)** *The set of* free agents/variables *of a* GSL *formula is given by the function* $\mathsf{free} : \mathrm{GSL} \to 2^{\mathrm{Ag} \cup \mathrm{Vr}}$ *such that:*

1. $\mathsf{free}(p) \triangleq \mathrm{Ag}$, *with* $p \in \mathrm{AP}$;
2. $\mathsf{free}(\neg\varphi) \triangleq \mathsf{free}(\varphi)$;
3. $\mathsf{free}(\varphi_1 \vee \varphi_2) \triangleq \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$;
4. $\mathsf{free}(\varphi_1 \wedge \varphi_2) \triangleq \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$;

5. $\mathsf{free}(\mathtt{X}\varphi) \triangleq \mathrm{Ag} \cup \mathsf{free}(\varphi)$;
6. $\mathsf{free}(\varphi_1 \mathtt{Op} \varphi_2) \triangleq \mathrm{Ag} \cup \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$ *with* $\mathtt{Op} \in \{\mathtt{U}, \mathtt{R}\}$;
7. $\mathsf{free}(\langle\!\langle x \geq g \rangle\!\rangle \varphi) \triangleq \mathsf{free}(\varphi) \setminus \{x\}$;
8. $\mathsf{free}([\![x < g]\!]\varphi) \triangleq \mathsf{free}(\varphi) \setminus \{x\}$;
9. $\mathsf{free}((a, x)\varphi) \triangleq \mathsf{free}(\varphi)$, *if* $a \notin \mathsf{free}(\varphi)$, *with* $a \in \mathrm{Ag}$ *and* $x \in \mathrm{Vr}$;
10. $\mathsf{free}((a, x)\varphi) \triangleq (\mathsf{free}(\varphi) \setminus \{a\}) \cup \{x\}$, *if* $a \in \mathsf{free}(\varphi)$, *with* $a \in \mathrm{Ag}$ *and* $x \in \mathrm{Vr}$.

A formula φ without free agents (resp., variables), *i.e.*, with $\mathsf{free}(\varphi) \cap \mathrm{Ag} = \emptyset$ (resp., $\mathsf{free}(\varphi) \cap \mathrm{Vr} = \emptyset$)), is named *agent-closed* (resp., *variable-closed*). A *sentence* is a both agent- and variable-closed formula. Since a variable x may be bound to more than a single agent at the time, we also need the subset $\mathsf{shr}(\varphi, x)$ of Ag containing those agents for which a binding (a, x) occurs in φ.

Definition 3.2.4 (GSL **Shared Variables)** *The set of* shared variables *of a* GSL *formula is given by the function* $\mathsf{shr} : \mathrm{GSL} \times \mathrm{Vr} \to 2^{\mathrm{Ag}}$ *such that:*

1. $\mathsf{shr}(p, x) \triangleq \emptyset$, *with* $p \in \mathrm{AP}$;
2. $\mathsf{shr}(\neg\varphi, x) \triangleq \mathsf{shr}(\varphi, x)$;
3. $\mathsf{shr}(\varphi_1 \vee \varphi_2, x) \triangleq \mathsf{shr}(\varphi_1, x) \cup \mathsf{shr}(\varphi_2, x)$;
4. $\mathsf{shr}(\varphi_1 \wedge \varphi_2, x) \triangleq \mathsf{shr}(\varphi_1, x) \cup \mathsf{shr}(\varphi_2, x)$;
5. $\mathsf{shr}(\mathtt{X}\varphi, x) \triangleq \mathsf{shr}(\varphi, x)$;
6. $\mathsf{shr}(\varphi_1 \mathtt{Op} \varphi_2, x) \triangleq \mathsf{shr}(\varphi_1, x) \cup \mathsf{shr}(\varphi_2, x)$ *with* $\mathtt{Op} \in \{\mathtt{U}, \mathtt{R}\}$;
7. $\mathsf{shr}(\langle\!\langle x \geq g \rangle\!\rangle \varphi, x) \triangleq \mathsf{shr}(\varphi, x)$;
8. $\mathsf{shr}([\![x < g]\!]\varphi, x) \triangleq \mathsf{shr}(\varphi, x)$;
9. $\mathsf{shr}((a, y)\varphi, x) \triangleq \mathsf{shr}(\varphi, x)$, *if* $a \notin \mathsf{free}(\varphi)$ *or* $y \neq x$, *with* $a \in \mathrm{Ag}$ *and* $y \in \mathrm{Vr}$;
10. $\mathsf{shr}((a, x)\varphi, x) \triangleq \mathsf{shr}(\varphi, x) \cup \{a\}$, *if* $a \in \mathsf{free}(\varphi)$, *with* $a \in \mathrm{Ag}$.

For complexity reasons, we restrict to the *One-Goal* fragment of GSL (GSL[1G], for short), which is the graded extension of SL[1G] [MMPV14]. To formalize its syntax, we first introduce some notions. A *quantification prefix* over a set $\mathrm{V} \subseteq \mathrm{Vr}$ of variables is a word $\wp \in \{\langle\!\langle x \geq g \rangle\!\rangle, [\![x < g]\!] : x \in \mathrm{V} \wedge g \in \mathbb{N}\}^{|\mathrm{V}|}$ of length $|\mathrm{V}|$ such that each $x \in \mathrm{V}$ occurs just once in $\wp$. With $\mathrm{Qn}(V)$ we indicate the set of quantification prefixes over V. A *binding prefix* over $\mathrm{A} \subseteq \mathrm{Ag}$ is a word $\flat \in \{(a, x) : a \in \mathrm{A} \wedge x \in \mathrm{Vr}\}^{|\mathrm{A}|}$ such that each $a \in \mathrm{A}$ occurs exactly once

in $\flat$. By Bn we indicate the set of all binding prefixes. GSL[1G] restricts GSL by forcing, after a quantification prefix, a single goal to occur *i.e.*, a formula of the kind $\flat\psi$, where $\flat$ is a binding prefix on all the agents in Ag. The syntax of GSL[1G] follows.

Definition 3.2.5 (GSL[1G] **Syntax)** GSL[1G] formulas *are built inductively through the following grammar:*

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathtt{R} \varphi \mid \wp\flat\varphi,$$

with $\wp$ quantification prefix over $\mathsf{free}(\flat\varphi)$ *and $\flat\varphi$ a goal.*

As an example of GSL[1G] property, in the context of the scheduler system, consider the sentence $\varphi = \wp\flat\psi$, with $\wp = \langle\langle \mathtt{x} \geq k \rangle\rangle [\![\mathtt{y}_1 < g_1]\!] [\![\mathtt{y}_2 < g_2]\!]$, $\flat = (\mathtt{A}, \mathtt{x})(\mathtt{P}_1, \mathtt{y}_1)(\mathtt{P}_2, \mathtt{y}_2)$, and $\psi = \mathtt{F}(\mathtt{r}_1 \vee \mathtt{r}_2)$. It says that there are at least k strategies for the arbiter $\mathtt{A}$ ensuring that one between the processes $\mathtt{P}_1$ and $\mathtt{P}_2$ receives the resource, being them able to avoid less than g_1 and g_2 strategies, respectively.

3.2.3 Semantics

As for SL, the interpretation of a GSL formula requires a valuation for its free placeholders. This is done via *assignments*, *i.e.*, partial functions $\chi \in \mathrm{Asg} \triangleq (\mathrm{Vr} \cup \mathrm{Ag}) \rightharpoonup \mathrm{Str}$ mapping variables/agents to strategies. An assignment χ is *complete* if it is defined on all agents in Ag, *i.e.*, $\chi(a) \in \mathrm{Str}(\{a\})$, for all $a \in \mathrm{Ag} \subseteq \mathsf{dom}(\chi)$. In this case, it directly identifies the profile $\chi_{\restriction \mathrm{Ag}}$ given by the restriction of χ to Ag. In addition, $\chi[e \mapsto \sigma]$, with $e \in \mathrm{Vr} \cup \mathrm{Ag}$ and $\sigma \in \mathrm{Str}$, denotes the assignment defined on $\mathsf{dom}(\chi[e \mapsto \sigma]) \triangleq \mathsf{dom}(\chi) \cup \{e\}$ that differs from χ only on the fact that e is associated with σ. Formally, $\chi[e \mapsto \sigma](e) = \sigma$ and $\chi[e \mapsto \sigma](e') = \chi(e')$, for all $e' \in \mathsf{dom}(\chi) \setminus \{e\}$. For a state $s \in \mathrm{St}$, it is said that χ is *s-total* if all strategies $\chi(l)$ are *s-total*, for $l \in \mathsf{dom}(\chi)$. The set $\mathrm{Asg} \triangleq \mathrm{Vr} \cup \mathrm{Ag} \rightharpoonup \mathrm{Str}$ (resp., $\mathrm{Asg}(s) \triangleq \mathrm{Vr} \cup \mathrm{Ag} \rightharpoonup \mathrm{Str}(s)$) contains all (resp., s-total) assignments. Moreover, $\mathrm{Asg}(\mathrm{X}) \triangleq \mathrm{X} \rightharpoonup \mathrm{Str}$ (resp., $\mathrm{Asg}(\mathrm{X}, s) \triangleq \mathrm{X} \rightharpoonup \mathrm{Str}(s)$) indicates the subset of X-defined (resp., s-total) assignments, *i.e.*, (resp., s-total) assignments defined on the set $\mathrm{X} \subseteq \mathrm{Vr} \cup \mathrm{Ag}$. Finally, for a formula φ, we say that χ is *φ-coherent* iff *(i)* $\mathsf{free}(\varphi) \subseteq \mathsf{dom}(\chi)$, *(ii)* $\chi(a) \in \mathrm{Str}(\{a\})$, for all $a \in \mathsf{dom}(\chi) \cap \mathrm{Ag}$, and *(iii)* $\chi(x) \in \mathrm{Str}(\mathsf{shr}(\varphi, x))$, for all $x \in \mathsf{dom}(\chi) \cap \mathrm{Vr}$. To provide the semantics of GSL, we give a definition of *update state/assignment* which is used to calculate, at a certain step of the play, what the current state and its updated assignment are. For a given state $s \in \mathrm{St}$ and a complete s-total assignment $\chi \in \mathrm{Asg}(s)$, the i-th update state/assignment of (χ, s), with $i \in \mathbb{N}$, is the pair of a complete assignment and a state $(\chi, s)^i \triangleq ((\chi)_{(\pi)_{\leq i}}, (\pi)_i)$ where $\pi = \mathsf{play}(\chi, s)$. In other words, $(\chi, s)^i$ corresponds to the i-th element of the sequence, given a partial assignments $(\chi)_{(\pi)_{\leq i}}$.

We now define the semantics of a GSL formula φ *w.r.t.* a CGS $\mathcal{G}$ and a φ-coherent assignment χ. In particular, we write $\mathcal{G}, \chi \models \varphi$ to indicate that φ holds in $\mathcal{G}$ under χ. The semantics of LTL formulas and agent bindings are defined as in SL. The definition of graded strategy quantifiers, instead, makes use of a family of equivalence relations $\equiv^{\varphi}_{\mathcal{G}}$ on assignments that depend on the structure $\mathcal{G}$ and the considered formula φ. This equivalence is used to reasonably count the number of strategies that satisfy a formula *w.r.t.* an *a priori* fixed criterion. Observe that we use a relation on assignments instead of a more direct one on strategies, since the classification may also depend on the context determined by the strategies previously quantified. In Section 3.3, we will come back to the properties the equivalence relation has to satisfy in order to be used in the semantics of GSL.

Definition 3.2.6 (GSL **Semantics)** *Let $\mathcal{G}$ be a* CGS*, φ be a* GSL *formula and $s \in \mathrm{St}$ be a state. For all φ-coherent assignments $\chi \in \mathrm{Asg}$, the relation $\mathcal{G}, \chi, s \models \varphi$ is inductively defined as follows.*

1. *For every $p \in \mathrm{AP}$, it holds that $\mathcal{G}, \chi, s \models p$ iff $p \in \mathsf{ap}(s)$.*
2. *For all formulas φ, φ_1, and φ_2, it holds that:*
   (a) *$\mathcal{G}, \chi, s \models \neg\varphi$ iff $\mathcal{G}, \chi, s \not\models \varphi$;*
   (b) *$\mathcal{G}, \chi, s \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{G}, \chi, s \models \varphi_1$ and $\mathcal{G}, \chi, s \models \varphi_2$;*
   (c) *$\mathcal{G}, \chi, s \models \varphi_1 \vee \varphi_2$ iff $\mathcal{G}, \chi, s \models \varphi_1$ or $\mathcal{G}, \chi, s \models \varphi_2$.*
3. *For each $x \in \mathrm{Vr}$, $g \in \mathbb{N}$, and $\varphi \in$ GSL, it holds that:*
   (a) *$\mathcal{G}, \chi, s \models \langle\!\langle x \geq g \rangle\!\rangle \varphi$ iff $|(\{\chi[x \mapsto \sigma] : \sigma \in \varphi[\mathcal{G}, \chi, s](x)\}/\equiv^{\varphi}_{\mathcal{G}})| \geq g$;*
   (b) *$\mathcal{G}, \chi, s \models [\![x < g]\!] \varphi$ iff $|(\{\chi[x \mapsto \sigma] : \sigma \in \neg\varphi[\mathcal{G}, \chi, s](x)\}/\equiv^{\neg\varphi}_{\mathcal{G}})| < g$;*

   where $\eta[\mathcal{G}, \chi, s](x) \triangleq \{\sigma \in \mathrm{Str}(\mathsf{shr}(\eta, x)) : \mathcal{G}, \chi[x \mapsto \sigma], s \models \eta\}$ is the set of $\mathsf{shr}(\eta, x)$-coherent strategies that, being assigned to x in χ, satisfy η.
4. *For each $a \in \mathrm{Ag}$, $x \in \mathrm{Vr}$, and $\varphi \in$ GSL, it holds that $\mathcal{G}, \chi, s \models (a, x)\varphi$ iff $\mathcal{G}, \chi[a \mapsto \chi(x)], s \models \varphi$.*
5. *Finally, if the assignment χ is also complete, for all formulas φ, φ_1, and φ_2, it holds that:*
   (a) *$\mathcal{G}, \chi, s \models \mathtt{X}\varphi$ iff $\mathcal{G}, (\chi, s)^1 \models \varphi$;*
   (b) *$\mathcal{G}, \chi, s \models \varphi_1 \mathtt{U} \varphi_2$ if there is an index $i \in \mathbb{N}$ such that $\mathcal{G}, (\chi, s)^i \models \varphi_2$ and, for all indexes $j \in \mathbb{N}$ with $j < i$, it holds that $\mathcal{G}, (\chi, s)^j \models \varphi_1$;*
   (c) *$\mathcal{G}, \chi, s \models \varphi_1 \mathsf{R} \varphi_2$ if, for all indexes $i \in \mathbb{N}$, it holds that $\mathcal{G}, (\chi, s)^i \models \varphi_2$ or, there is an index $j \in \mathbb{N}$ with $j < i$, such that $\mathcal{G}, (\chi, s)^j \models \varphi_1$.*

Intuitively, the existential quantifier $\langle\!\langle x \geq g \rangle\!\rangle \varphi$ allows us to count the number of equivalence classes *w.r.t.* $\equiv_{\mathcal{G}}^{\varphi}$ over the set of assignments $\{\chi[x \mapsto \sigma] : \sigma \in \varphi[\mathcal{G}, \chi](x)\}$ that, extending χ, satisfy φ. The universal quantifier $[\![x < g]\!]\varphi$ is the dual of $\langle\!\langle x \geq g \rangle\!\rangle \varphi$ and counts how many classes *w.r.t.* $\equiv_{\mathcal{G}}^{\neg\varphi}$ there are over the assignments $\{\chi[x \mapsto \sigma] : \sigma \in \neg\varphi[\mathcal{G}, \chi](x)\}$ that, extending χ, do not satisfy φ. Note that all GSL formulas with degree 1 are SL formulas, since with $\langle\!\langle \mathtt{x} \geq 1 \rangle\!\rangle \varphi$ is appropriate to find a single strategy that satisfies the formula, just like $\langle\!\langle \mathtt{x} \rangle\!\rangle \varphi$. Furthermore, by $[\![\mathtt{x} < 1]\!]\varphi$ all strategies are considered, without excluding any, just like $[\![\mathtt{x}]\!]\varphi$. In order to complete the description of the semantics, we now give the classic notions of model and satisfiability of an GSL sentence. We say that a CGS $\mathcal{G}$ is a model of an GSL sentence φ, in symbols $\mathcal{G} \models \varphi$, if $\mathcal{G}, \varnothing, s_I \models \varphi$.[2] In general, we also say that $\mathcal{G}$ is a model for φ on $s \in \mathrm{St}$, in symbols $\mathcal{G}, s \models \varphi$, if $\mathcal{G}, \varnothing, s \models \varphi$. An GSL sentence φ is satisfiable if there is a model for it.

Consider again the sentence $\varphi = \langle\!\langle \mathtt{x} \geq k \rangle\!\rangle [\![\mathtt{y}_1 < g_1]\!] [\![\mathtt{y}_2 < g_2]\!] (\mathtt{A}, \mathtt{x})(\mathtt{P}_1, \mathtt{y}_1)(\mathtt{P}_2, \mathtt{y}_2)$ $\mathtt{F}(\mathtt{r}_1 \vee \mathtt{r}_2)$ of the scheduler example. Once a reasonable equivalence relation on assignments is fixed (see Section 3.3), one can see that $\mathcal{G}_S \models \varphi$ with $k \geq 0$ and $(g_1, g_2) = (1, 2)$ but $\mathcal{G}_S \not\models \varphi$ with $(k, g_1, g_2) = (1, 1, 1)$. Indeed, if the processes use the same strategy, they may force the play to be in $(\mathtt{I}^+ \cdot \mathtt{W})^* \cdot \mathtt{I}^\omega + (\mathtt{I}^+ \cdot \mathtt{W})^\omega$, so they either avoid to do a request or relinquish a request that is not immediately served. Consequently, to satisfy φ, we need to verify the property against all but one strategy of $\mathtt{P}_2$, *i.e.*, the one used by $\mathtt{P}_1$. Under these assumptions, we can see that the arbiter $\mathtt{A}$ has an infinite number of different strategies by suitably choosing the actions on all histories ending in the state $\mathtt{W}$.

3.2.4 Results

In this section, we summarize the main results we have obtained on GSL along this paper. We start showing that graded ATL (GATL) [FNP09a] is strictly included in GSL[1G]. Precisely, we first show that GATL can be translated into GSL[1G], then we provide a formula GSL[1G] and show that it cannot be expressed in GATL. In [FNP09a], the authors introduce two different semantics for GATL, called *off-line* and *on-line*, and it has been showed that this logic has the ability to count how many different strategies (in the off-line semantics) or paths (in the on-line semantics) satisfy a certain property. Under the off-line semantics, over a CGS with agents α and $\overline{\alpha}$, the GATL formula $\langle\!\langle \alpha \rangle\!\rangle^g \psi$ is equivalent to the GSL[1G] sentence $\langle\!\langle \mathtt{x} \geq g \rangle\!\rangle [\![\overline{\mathtt{x}} < 1]\!] (\alpha, \mathtt{x})(\overline{\alpha}, \overline{\mathtt{x}})\psi$. Under the on-line semantics, instead, it is equivalent to the sentence $[\![\overline{\mathtt{x}} < 1]\!] \langle\!\langle \mathtt{x} \geq g \rangle\!\rangle (\alpha, \mathtt{x})(\overline{\alpha}, \overline{\mathtt{x}})\psi$. Note that the counting over strategies in GATL is limited to existential agents and, so, the GSL[1G] formula $[\![\overline{\mathtt{x}} < 2]\!] \langle\!\langle \mathtt{y} \geq 1 \rangle\!\rangle (\alpha, \mathtt{x})(\overline{\alpha}, \mathtt{y})\psi$ does not have any ATL equivalent formula. From these considerations, we derive the following theorem.

[2] The symbol $\varnothing$ stands for the empty function.

Theorem 3.2.1 GSL[1G] *is more expressive of* GATL.

It is important to note that the criteria used for the strategy classification in GATL is strictly coupled with the temporal operators $\mathtt{X}\varphi$, $\varphi_1 \mathtt{U} \varphi_2$, and $\mathtt{G}\varphi$ along the syntax, and we do not see how this can be extended to the whole LTL, unless one uses the approach proposed in [BMM12].

Another important result we prove in Section 3.5 is the determinacy for GSL[1G] in the case of 2 variables as stated in the following theorem.

Theorem 3.2.2 (Determinacy) GSL[1G, 2AG] *on Turn-Based Game Structures is determined.*

Finally, in Section 3.6, we solve the model checking problem for the vanilla fragment of GSL[1G] with 2 variables. As for ATL, *Vanilla* GSL[1G] requires that two successive temporal operators in a formula are always interleaved by a strategy quantifier.

Theorem 3.2.3 (Model Checking) *The model-checking problem for* Vanilla GSL[1G] *is* PTIME-COMPLETE w.r.t. *the size of the structure and the sentence.*

3.3 Strategy Equivalence

Our definition of the GSL semantics makes use of an arbitrary family of equivalence relation on assignments. This choice introduces flexibility in its description, since one can come up with different logics by opportunely choosing different equivalences. In this section, we focus on a particular relation whose key feature is to classify as equivalent all assignments that reflect the same "*strategic reasoning*", although they may have completely different structures. Just to get an intuition about what we mean, consider two assignments χ_1 and χ_2 and the corresponding involved strategies associated with the agents $\mathtt{a}_1$ and $\mathtt{a}_2$. Assume now that, for each $i \in \{1, 2\}$, the homologous strategies $\chi_1(\mathtt{a}_i)$ and $\chi_2(\mathtt{a}_i)$ only differ on histories never met by a play because of a specific combination of their actions. Clearly, χ_1 and χ_2 induce the same agent behaviors, which means to reflect the same strategic reasoning. Therefore, it is natural to set them as equivalent, as we do. Two formulas are considered equivalent whenever the two assignments are equivalent for both or none of them. Also, if two assignments do not satisfy the same formulas, they are not equivalent.

In the sequel, in order to illustrate the introduced concepts, we analyze subformulas of the previously described sentence $\langle\!\langle \mathtt{x} \geq k \rangle\!\rangle [\![\mathtt{y}_1 < 1]\!] [\![\mathtt{y}_2 < 2]\!] (\mathtt{A},\mathtt{x})(\mathtt{P}_1,\mathtt{y}_1)(\mathtt{P}_2,\mathtt{y}_2)\mathtt{F}(\mathtt{r}_1 \vee \mathtt{r}_2)$, together with their negations, over the CGS $\mathcal{G}_S$ of Figure 3.1.

3.3.1 Elementary Requirements

Logics usually admit syntactic redundancy. For example, in LTL we have $\neg\mathtt{X}(\mathtt{p} \wedge \mathtt{q}) \equiv \mathtt{X}\neg(\mathtt{p} \wedge \mathtt{q}) \equiv \mathtt{X}(\neg\mathtt{p} \vee \neg\mathtt{q})$. Also, the semantics is normally closed under substitution. Yet for LTL, this means that $\neg\mathtt{X}(\mathtt{p} \wedge \mathtt{q})$ can be replaced with $\mathtt{X}\neg(\mathtt{p} \wedge \mathtt{q})$ or $\mathtt{X}(\neg\mathtt{p} \vee \neg\mathtt{q})$, without changing the meaning of a formula. GSL should not be an exception. To ensure this, we require the invariance of the equivalence relation on assignments *w.r.t.* the syntax of the involved formulas.

Definition 3.3.1 (Syntax Independence) *An equivalence relation on assignments $\equiv^{\cdot}_{\mathcal{G}}$ is* syntax independent *if, for any pair of equivalent formulas φ_1 and φ_2 and $(\mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2))$-coherent assignments $\chi_1, \chi_2 \in \mathrm{Asg}$, we have that $\chi_1 \equiv^{\varphi_1}_{\mathcal{G}} \chi_2$ iff $\chi_1 \equiv^{\varphi_2}_{\mathcal{G}} \chi_2$.*

As declared above, our aim is to classify as equivalent *w.r.t.* a formula φ all assignments that induce the same strategic reasoning. Therefore, we cannot distinguish them *w.r.t.* the satisfiability of φ itself.

Definition 3.3.2 (Semantic Consistency) *An equivalence relation on assignments $\equiv^{\cdot}_{\mathcal{G}}$ is* semantically consistent *if, for any formula φ and φ-coherent assignments $\chi_1, \chi_2 \in \mathrm{Asg}$, we have that if $\chi_1 \equiv^{\varphi}_{\mathcal{G}} \chi_2$ then either $\mathcal{G}, \chi_1 \models \varphi$ and $\mathcal{G}, \chi_2 \models \varphi$ or $\mathcal{G}, \chi_1 \not\models \varphi$ and $\mathcal{G}, \chi_2 \not\models \varphi$.*

3.3.2 Play Requirement

We now deal with the equivalence relation for the basic case of temporal properties. Before disclosing the formalization, we give an intuition on how to evaluate the equivalence of two complete assignments χ_1 and χ_2 *w.r.t.* their agreement on the verification of a generic LTL property ψ. Let π_1 and π_2 with $\pi_1 \neq \pi_2$ be the plays satisfying ψ induced by χ_1 and χ_2, respectively. Also, consider their maximal common prefix $\rho = \mathsf{prf}(\pi_1, \pi_2) \in \mathrm{Hst}$. If ρ can be extended to a play in such a way that ψ does not hold, we are sure that the reasons why both the assignments satisfy the property are different, as they reside in the parts where the two plays diverge. Consequently, we can assume χ_1 and χ_2 to be non-equivalent *w.r.t.* ψ. Conversely, if all infinite extensions of ρ necessarily satisfy ψ, we may affirm that this is already a witness of the verification of the property by the two plays and, so, by the two assignments. Hence, we can assume χ_1 and χ_2 to be equivalent *w.r.t.* ψ.

In the following, we often make use of the concept of witness of an LTL formula ψ as the set $\mathrm{W}_\psi \triangleq \{\rho \in \mathrm{Hst} : \forall \pi \in \mathrm{Pth} \,.\, \rho < \pi \Rightarrow \pi \models \psi\}$ containing all histories that cannot be extended to a play violating the property.

Definition 3.3.3 (Play Consistency) *An equivalence relation on assignments $\equiv^{\cdot}_{\mathcal{G}}$ is* play consistent *if, for any* LTL *formula ψ and ψ-coherent assignments $\chi_1, \chi_2 \in \mathrm{Asg}$, we have that $\chi_1 \equiv^{\psi}_{\mathcal{G}} \chi_2$ iff either $\pi_1 = \pi_2$ or $\mathsf{prf}(\pi_1, \pi_2) \in \mathrm{W}_\psi$, where $\pi_1 = \mathsf{play}(\chi_{1\restriction \mathrm{Ag}}, s_I)$ and*

$\pi_2 = \mathsf{play}(\chi_{2\restriction \mathrm{Ag}}, s_I)$ *are the plays induced by* χ_1 *and* χ_2*, respectively, and* $\mathrm{W}_\psi \subseteq \mathrm{Hst}$ *is the witness set of* ψ*.*

To see how to apply the above definition, consider the formula $\psi = \mathtt{F}(\mathtt{r}_1 \vee \mathtt{r}_2)$ and let W_ψ be the corresponding witness set, whose minimal histories can be represented by the regular expression $\mathtt{I}^+ \cdot (1 + 2) + (\mathtt{I}^+ \cdot \mathtt{W})^+ \cdot (1 + 2 + 1/2 + 2/1)$. Moreover, let $\chi_1, \chi_2, \chi_3 \in \mathrm{Asg}(\{\mathtt{A}, \mathtt{P}_1, \mathtt{P}_2\})$ be three complete assignments on which we want to check the play consistency. We assume that each χ_i associates a strategy $\chi_i(\mathtt{a}) = \sigma_i^{\mathtt{a}}$ with the agent $\mathtt{a} \in \{\mathtt{A}, \mathtt{P}_1, \mathtt{P}_2\}$ as defined in the following, where $\rho, \rho' \in \mathrm{Hst}$ with $\mathsf{lst}(\rho') \neq \mathtt{I}$: for the arbiter $\mathtt{A}$, we set $\sigma^{\mathtt{A}}_{1/2}(\rho \cdot \mathtt{W}) \triangleq 2$, $\sigma^{\mathtt{A}}_{1/2/3}(\rho \cdot 1/2) = \sigma^{\mathtt{A}}_{2}(\rho \cdot 2/1) \triangleq \mathtt{i}$, and $\sigma^{\mathtt{A}}_{3}(\rho \cdot \mathtt{W}) = \sigma^{\mathtt{A}}_{1/3}(\rho \cdot 2/1) \triangleq 1$; for the processes, instead, we set $\sigma^{\mathtt{P}_1}_{1/2/3}(\rho') = \sigma^{\mathtt{P}_2}_{1/2/3}(\rho') \triangleq \mathtt{i}$, $\sigma^{\mathtt{P}_1}_{1/2}(\rho \cdot \mathtt{I}) = \sigma^{\mathtt{P}_2}_{1/2/3}(\rho \cdot \mathtt{I}) \triangleq \mathtt{r}$, and $\sigma^{\mathtt{P}_1}_{3}(\rho \cdot \mathtt{I}) \triangleq \mathtt{i}$[3]. Now, one can see that $\chi_1 \equiv^{\psi}_{\mathcal{G}} \chi_2$, but $\chi_1 \not\equiv^{\psi}_{\mathcal{G}} \chi_3$. Indeed, χ_1, χ_2, and χ_3 induce the plays $\pi_1 = \mathtt{I} \cdot \mathtt{W} \cdot 2/1 \cdot 1/2^\omega$, $\pi_2 = \mathtt{I} \cdot \mathtt{W} \cdot 2/1^\omega$, and $\pi_3 = \mathtt{I} \cdot 2^\omega$, respectively, where $\rho_{12} = \mathsf{prf}(\pi_1, \pi_2) = \mathtt{I} \cdot \mathtt{W} \cdot 2/1$ and $\rho_{13} = \mathsf{prf}(\pi_1, \pi_3) = \mathtt{I}$ are the corresponding common prefixes. Thus, ρ_{12} belongs to the witness W_ψ, while ρ_{13} does not.

As another example, consider the formula $\overline{\psi} = \mathtt{G}(\neg \mathtt{r}_1 \wedge \neg \mathtt{r}_2)$, which is equivalent to the negation of the previous one, and observe that its witness set $\mathrm{W}_{\overline{\psi}}$ is empty. Moreover, let $\chi_1, \chi_2, \chi_3 \in \mathrm{Asg}(\{\mathtt{A}, \mathtt{P}_1, \mathtt{P}_2\})$ be the three complete assignments we want to analyze. The strategies for the arbiter $\mathtt{A}$ are defined as above, while those of the processes follows: $\overline{\sigma}^{\mathtt{P}_i}_{1/2/3}(\rho') \triangleq \mathtt{i}$, $\overline{\sigma}^{\mathtt{P}_i}_{1/2}(\rho \cdot \mathtt{I}) \triangleq \mathtt{r}$, $\overline{\sigma}^{\mathtt{P}_i}_{1/2}(\rho \cdot \mathtt{W}) \triangleq \mathtt{a}$, and $\overline{\sigma}^{\mathtt{P}_i}_{3}(\rho \cdot \mathtt{I}) = \overline{\sigma}^{\mathtt{P}_i}_{3}(\rho \cdot \mathtt{W}) \triangleq \mathtt{i}$, where $i \in \{1, 2\}$ and $\rho, \rho' \in \mathrm{Hst}$ with $\mathsf{lst}(\rho') \notin \{\mathtt{I}, \mathtt{W}\}$. Now, one can see that $\chi_1 \equiv^{\overline{\psi}}_{\mathcal{G}} \chi_2$, but $\chi_1 \not\equiv^{\overline{\psi}}_{\mathcal{G}} \chi_3$. Indeed, χ_1 and χ_2 induce the same play $(\mathtt{I} \cdot \mathtt{W})^\omega$, while χ_3 runs along $\mathtt{I}^\omega$. Thus, χ_1 and χ_2 are equivalent, but χ_1 and χ_3 are not.

3.3.3 Strategy Requirements

The semantics of a binding construct $\varphi = (a, x)\eta$ involves a redefinition of the underlying assignment χ, as it asserts that φ holds under χ once the inner part η is satisfied by associating the agent a to the strategy $\chi(x)$. Thus, the equivalence of two assignments χ_1 and χ_2 *w.r.t.* φ necessarily depends on that of their extensions on a *w.r.t.* η.

Definition 3.3.4 (Binding Consistency) *An equivalence relation on assignments* $\equiv^{\cdot}_{\mathcal{G}}$ *is* binding consistent *if, for a formula* $\varphi = (\mathtt{a}, \mathtt{x})\eta$ *and* φ*-coherent assignments* $\chi_1, \chi_2 \in \mathrm{Asg}$*, we have that* $\chi_1 \equiv^{\varphi}_{\mathcal{G}} \chi_2$ *iff* $\chi_1[\mathtt{a} \mapsto \chi_1(\mathtt{x})] \equiv^{\eta}_{\mathcal{G}} \chi_2[\mathtt{a} \mapsto \chi_2(\mathtt{x})]$.

To get familiar with the above concept, consider the formula $\flat\psi$, where $\flat \triangleq (\mathtt{A}, \mathtt{x})(\mathtt{P}_1, \mathtt{y}_1)(\mathtt{P}_2, \mathtt{y}_2)$, and let $\chi_1, \chi_2, \chi_3 \in \mathrm{Asg}(\{\mathtt{x}, \mathtt{y}_1, \mathtt{y}_2\})$ be the assignments assuming as values the strategies $\chi_i(\mathtt{x}) \triangleq \sigma_i^{\mathtt{A}}$ and $\chi_i(\mathtt{y}_j) \triangleq \sigma_i^{\mathtt{P}_j}$ previously defined, where $i \in \{1, 2, 3\}$ and $j \in \{1, 2\}$. Then, by definition, it is immediate to see that $\chi_1 \equiv^{\flat\psi}_{\mathcal{G}} \chi_2$, but $\chi_1 \not\equiv^{\flat\psi}_{\mathcal{G}} \chi_3$.

[3]Note that, we use $\sigma^{\mathtt{a}}_{i/j}(\rho) = \alpha$ to represent $\sigma^{\mathtt{a}}_{i}(\rho) = \alpha$ and $\sigma^{\mathtt{a}}_{j}(\rho) = \alpha$.

Before continuing with the analysis of the equivalence, it is worth making same reasoning about the dual nature of the existential and universal quantifiers *w.r.t.* the counting of strategies. We do this by exploiting the classic game-semantics metaphor originally proposed for first-order logic by Lorenzen and Hintikka, where the choice of an existential variable is done by a player called $\exists$ and that of the universal ones by its opponent $\forall$. Consider a sentence $\langle\!\langle x_1 \geq g_1 \rangle\!\rangle [\![x_2 < g_2]\!] \eta$, having $\langle\!\langle \mathtt{y}_1 \geq h_1 \rangle\!\rangle \eta_1$ and $[\![\mathtt{y}_2 < h_2]\!] \eta_2$ as two subformulas in η. When player $\exists$ tries to choose h_1 different strategies $\mathtt{y}_1$ to satisfy η_1, it also has to maximize the number of strategies x_1 by verifying $[\![x_2 < g_2]\!] \eta$ to be sure that the constraint $\geq g_1$ of the first quantification is not violated. At the same time, player $\forall$ tries to do the opposite while choosing h_2 different strategies y_2 not satisfying η_2, *i.e.*, it needs to maximize the number of strategies x_2 falsifying η in order to violate the constraint $< g_2$ of the second quantifier.

With the above observation in mind, we now treat the equivalence for the existential quantifier. Two assignments χ_1 and χ_2 are equivalent *w.r.t.* a formula $\varphi = \langle\!\langle x \geq g \rangle\!\rangle \eta$ if player $\exists$ is not able to find a strategy σ among those satisfying η, to associate with the variable x, that allows the corresponding extensions of χ_1 and χ_2 on x to induce different behaviors *w.r.t.* η. In other words, $\exists$ cannot distinguish between the two assignments, as they behave the same independently of the way they are extended.

Definition 3.3.5 (Existential Consistency) *An equivalence relation on assignments $\equiv^{\cdot}_{\mathcal{G}}$ is* existentially consistent *if, for any formula $\varphi = \langle\!\langle \mathtt{x} \geq g \rangle\!\rangle \eta$ and φ-coherent assignments $\chi_1, \chi_2 \in \mathrm{Asg}$, we have that $\chi_1 \equiv^{\varphi}_{\mathcal{G}} \chi_2$ iff, for each strategy $\sigma \in \eta[\mathcal{G}, \chi_1](\mathtt{x}) \cup \eta[\mathcal{G}, \chi_2](\mathtt{x})$, it holds that $\chi_1[\mathtt{x} \mapsto \sigma] \equiv^{\eta}_{\mathcal{G}} \chi_2[\mathtt{x} \mapsto \sigma]$.*

To clarify the above definition, consider the formula $\varphi = \langle\!\langle \mathtt{y}_2 \geq 2 \rangle\!\rangle \flat \overline{\psi}$ and let $\chi_1, \chi_2, \chi_3 \in \mathrm{Asg}(\{\mathtt{x}, \mathtt{y}_1\})$ be the three assignments having as values the strategies $\chi_i(\mathtt{x}) \triangleq \overline{\sigma}^{\mathtt{A}}_i$ and $\chi_i(\mathtt{y}_1) \triangleq \overline{\sigma}^{\mathtt{P}_1}_i$ previously defined, where $i \in \{1, 2, 3\}$. By a matter of calculation, one can see that $\chi_1 \equiv^{\varphi}_{\mathcal{G}} \chi_2$, but $\chi_1 \not\equiv^{\varphi}_{\mathcal{G}} \chi_3$. By definition, $\chi_1 \equiv^{\varphi}_{\mathcal{G}} \chi_2$ iff, for each strategy $\overline{\sigma} \in (\flat\overline{\psi})[\mathcal{G}, \chi_1](\mathtt{y}_2) \cup (\flat\overline{\psi})[\mathcal{G}, \chi_2](\mathtt{y}_2)$, it holds that $\chi_1[\mathtt{y}_2 \mapsto \overline{\sigma}] \equiv^{\flat\overline{\psi}}_{\mathcal{G}} \chi_2[\mathtt{y}_2 \mapsto \overline{\sigma}]$. Now, observe that the strategy $\overline{\sigma}^{\mathtt{P}_2}_1$ introduced above is the unique one that allows χ_1 and χ_2 to satisfy $\flat\overline{\psi}$ once extended on $\mathtt{y}_2$. At this point, we can easily show that $\chi_1[\mathtt{y}_2 \mapsto \overline{\sigma}^{\mathtt{P}_2}_1] \equiv^{\flat\overline{\psi}}_{\mathcal{G}} \chi_2[\mathtt{y}_2 \mapsto \overline{\sigma}^{\mathtt{P}_2}_1]$, as the derived complete assignments $\chi_1[\mathtt{y}_2 \mapsto \overline{\sigma}^{\mathtt{P}_2}_1] \circ \flat$ and $\chi_2[\mathtt{y}_2 \mapsto \overline{\sigma}^{\mathtt{P}_2}_1] \circ \flat$ induce the same play $(\mathtt{I} \cdot \mathtt{W})^{\omega}$. The non-equivalence of χ_1 and χ_3 easily follows from the fact that $\overline{\sigma}^{\mathtt{P}_2}_1 \notin (\flat\overline{\psi})[\mathcal{G}, \chi_3](\mathtt{y}_2)$, as $\chi_3[\mathtt{y}_2 \mapsto \overline{\sigma}^{\mathtt{P}_2}_1] \circ \flat$ induces the play $\mathtt{I} \cdot 2^{\omega}$ that does not satisfy $\overline{\psi}$. Thus, $\chi_1[\mathtt{y}_2 \mapsto \overline{\sigma}^{\mathtt{P}_2}_1] \not\equiv^{\flat\overline{\psi}}_{\mathcal{G}} \chi_3[\mathtt{y}_2 \mapsto \overline{\sigma}^{\mathtt{P}_2}_1]$.

We conclude with the equivalence for the universal quantifier. Two assignments χ_1 and χ_2 are equivalent *w.r.t.* a formula $\varphi = [\![x < g]\!] \eta$ if, for each index $i \in \{1, 2\}$ and strategy σ_i player $\forall$ chooses among those satisfying η under χ_i, there is a strategy σ_{3-i} this player can choose among those satisfying η under χ_{3-i} such that, once the two strategies are associated with the variable x, they make the corresponding extensions of assignments equivalent *w.r.t.*

η. This means that the parts of the concurrent game structure that are reachable under χ_1 and χ_2 contain exactly the same information *w.r.t.* the verification of the inner formula. In other words, $\forall$ cannot distinguish between the two assignments, as the induced subtrees of possible plays are practically the same.

Definition 3.3.6 (Universal Consistency) *An equivalence relation on assignments $\equiv^{\cdot}_{\mathcal{G}}$ is* universally consistent *if, for any formula $\varphi = [\![\mathtt{x} < g]\!]\eta$ and φ-coherent assignments $\chi_1, \chi_2 \in \mathrm{Asg}$, we have that $\chi_1 \equiv^{\varphi}_{\mathcal{G}} \chi_2$ iff, for all $i \in \{1,2\}$ and strategy $\sigma_i \in \eta[\mathcal{G}, \chi_i](\mathtt{x})$, there is a strategy $\sigma_{3-i} \in \eta[\mathcal{G}, \chi_{3-i}](\mathtt{x})$ such that $\chi_1[\mathtt{x} \mapsto \sigma_1] \equiv^{\eta}_{\mathcal{G}} \chi_2[\mathtt{x} \mapsto \sigma_2]$.*

Finally, to better understand the above definition, consider the formula $\varphi = [\![\mathtt{y}_1 < 1]\!]\eta$, where $\eta = [\![\mathtt{y}_2 < 2]\!]\flat\psi$, and let $\chi_1, \chi_2, \chi_3 \in \mathrm{Asg}(\{\mathtt{x}\})$ be the three assignments having as values the strategies $\chi_i(\mathtt{x}) \triangleq \sigma_i^{\mathtt{A}}$ previously defined, where $i \in \{1,2,3\}$. One can see that $\chi_1 \equiv^{\varphi}_{\mathcal{G}} \chi_2$, but $\chi_1 \not\equiv^{\varphi}_{\mathcal{G}} \chi_3$.

First, observe that $\eta[\mathcal{G}, \chi_1](\mathtt{y}_1) = \eta[\mathcal{G}, \chi_2](\mathtt{y}_1) = \mathrm{Str}$. Indeed, for all strategies $\sigma \in \mathrm{Str}$, we have that $\mathcal{G}, \chi_1[\mathtt{y}_1 \mapsto \sigma] \models \eta$ and $\mathcal{G}, \chi_2[\mathtt{y}_1 \mapsto \sigma] \models \eta$, since $\mathcal{G}, \chi_1[\mathtt{y}_1 \mapsto \sigma, \mathtt{y}_2 \mapsto \sigma'] \models \flat\psi$ and $\mathcal{G}, \chi_2[\mathtt{y}_1 \mapsto \sigma, \mathtt{y}_2 \mapsto \sigma'] \models \flat\psi$, for all $\sigma' \in \mathrm{Str}$ such that $\sigma \neq \sigma'$. This is due to the fact that the plays π_1 and π_2 induced by the two complete assignments $\chi_1[\mathtt{y}_1 \mapsto \sigma, \mathtt{y}_2 \mapsto \sigma'] \circ \flat$ and $\chi_2[\mathtt{y}_1 \mapsto \sigma, \mathtt{y}_2 \mapsto \sigma'] \circ \flat$ differ from $(\mathtt{I}^+ \cdot \mathtt{W})^* \cdot \mathtt{I}^\omega$ and $(\mathtt{I}^+ \cdot \mathtt{W})^\omega$, as the strategies of the two processes are different. Also, they share a common prefix $\rho = \mathsf{prf}(\pi_1, \pi_2)$ belonging to W_ψ, since the strategies of the arbiter only differ on the histories ending in the state $2/1$. We can now show that χ_1 and χ_2 are equivalent, by applying the above definition in which we assume that $\sigma_i = \sigma_{3-i}$.

To prove that χ_1 and χ_3 are non-equivalent, we show that there is a strategy $\sigma \in \eta[\mathcal{G}, \chi_1](\mathtt{y}_1)$ for χ_1 such that, for all strategies $\sigma' \in \eta[\mathcal{G}, \chi_3](\mathtt{y}_1)$ for χ_3, it holds that $\chi_1[\mathtt{y}_1 \mapsto \sigma] \not\equiv^{\eta}_{\mathcal{G}} \chi_3[\mathtt{y}_1 \mapsto \sigma']$. As before, observe that $\eta[\mathcal{G}, \chi_1](\mathtt{y}_1) = \eta[\mathcal{G}, \chi_3](\mathtt{y}_1) = \mathrm{Str}$ and choose $\sigma \in \mathrm{Str}$ as the strategy $\sigma_1^{\mathtt{P}_1}$ previously defined. At this point, one can easily see that all plays compatible with $\chi_1[\mathtt{y}_1 \mapsto \sigma] \circ \flat$ pass through either $\mathtt{I} \cdot 1$ or $\mathtt{I} \cdot \mathtt{W} \cdot 2/1$, while a play compatible with $\chi_3 \circ \flat$ cannot pass through the latter history. Thus, the non-equivalence of the two assignments immediately follows.

3.4 From Concurrent To Turn-Based Games

In this section, we transform a game in a simpler but equivalent form. Precisely, we show how to transform a game from concurrent to turn-based. The definition of the turn-based structure follows.

Definition 3.4.1 (Turn-Based Game Structure) *A* CGS $\mathcal{G}$ *is a* Turn-Based Game Structure *(*TBGS*, for short) if there exist a function* $\mathsf{own} : \mathrm{St} \to \mathrm{Ag}$*, named* owner function *such that,*

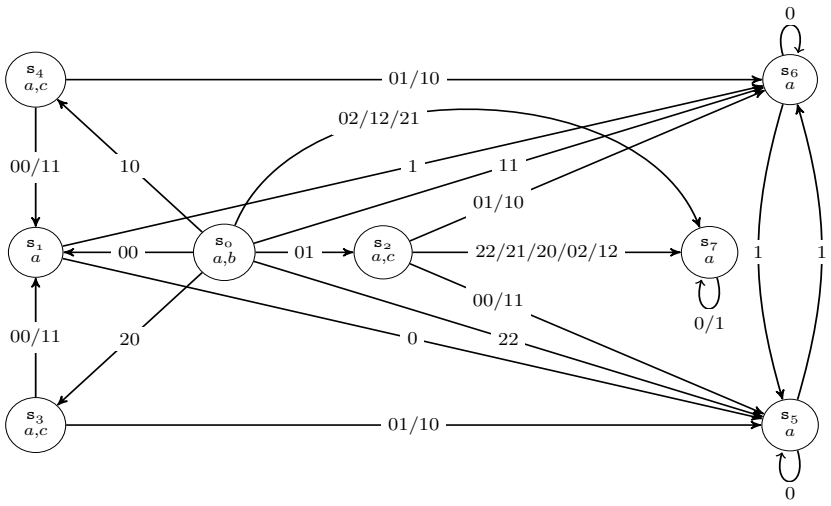


Figure 3.2: An example of CGS $\mathcal{G}$. Note that each node of $\mathcal{G}$ is labeled with its name (in the upper part) and the subset of the players that are active in it (in the lower part).

for all states $s \in \mathrm{St}$ *and decisions* $\delta_1, \delta_2 \in \mathrm{Dc}$*, it holds that if* $\delta_1(\mathsf{own}(s)) = \delta_2(\mathsf{own}(s))$ *then* $\mathsf{tr}(\delta_1)(s) = \mathsf{tr}(\delta_2)(s)$.

It is worth recalling that similar reductions have been also used to solve questions related to GATL in [FNP09a] and the one-goal fragment of SL in [MMS14b]. However, none of them can be used for GSL[1G]. The main reason resides in the fact that in both the mentioned cases, the reduction always results in a two-player game, where the two players represent a collapsing of all existential and universal modalities, respectively. Conversely, in GSL[1G] we need to maintain a multi-player setting in the construction. This is due to the fact that the technique employed in GSL[1G] to count the non-equivalent strategies in a quantification, say $\langle\langle \mathsf{x} \geq g \rangle\rangle \varphi$, depends on the particular kind of quantifications and counting on the variables contained in its matrix, *i.e.*, φ. In particular, it is worth recalling that in GATL strategies are grouped together w.r.t. set of agents, while in GSL[1G] every agent strategy is considered separately. Thus, we introduce an ad-hoc transformation of the concurrent game under exam into a multi-player turn-based one, which has the peculiarity of retaining the same number of variables, but can collapse equivalent actions. More precisely, starting with a game having k variables, we end in a game with k agents and k variables. The proposed conversion is divided into three parts. The first, called *normalization*, concerns the elimination of the bindings, where a different agent is introduced for every free variable. The second, named *minimization*, is the elimination of equivalent actions that are, therefore, redundant. Finally, the third is the real transformation of the game in a turn-based one. To better understand the three steps of the conversion, we consider the following running example.

Example 3.4.1 *Consider the* CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_{\mathrm{o}} \rangle$ *depicted in Figure 3.2,*

where $\mathrm{AP} = \{p\}$, $\mathrm{Ag} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, $\mathrm{Ac} = \{0, 1, 2\}$, $\mathrm{St} = \{\mathtt{s}_0, \mathtt{s}_1, \mathtt{s}_2, \mathtt{s}_3, \mathtt{s}_4, \mathtt{s}_5, \mathtt{s}_6, \mathtt{s}_7\}$, *and* $s_I = \mathtt{s}_0$. *Note that agent* $\mathtt{a}$ *is active in all states, agent* $\mathtt{b}$ *only in* $\mathtt{s}_0$, *and agent* $\mathtt{c}$ *in* $\mathtt{s}_2$, $\mathtt{s}_3$, *and* $\mathtt{s}_4$. *Moreover, we have that* $\mathsf{ac}(\mathtt{s}_0, \mathtt{a}) = \mathsf{ac}(\mathtt{s}_0, \mathtt{b}) = \mathsf{ac}(\mathtt{s}_2, \mathtt{a}) = \mathsf{ac}(\mathtt{s}_2, \mathtt{c}) = \{0, 1, 2\}$ *and* $\mathsf{ac}(\mathtt{s}_1, \mathtt{a}) = \mathsf{ac}(\mathtt{s}_1, \mathtt{c}) = \mathsf{ac}(\mathtt{s}_3, \mathtt{a}) = \mathsf{ac}(\mathtt{s}_3, \mathtt{c}) = \mathsf{ac}(\mathtt{s}_4, \mathtt{a}) = \mathsf{ac}(\mathtt{s}_4, \mathtt{c}) = \mathsf{ac}(\mathtt{s}_5, \mathtt{a}) = \mathsf{ac}(\mathtt{s}_6, \mathtt{a}) = \mathsf{ac}(\mathtt{s}_7, \mathtt{a}) = \{0, 1\}$. *Finally, the labeling function is defined as* $\mathsf{ap}(\mathtt{s}_0) = \mathsf{ap}(\mathtt{s}_2) = \mathsf{ap}(\mathtt{s}_3) = \mathsf{ap}(\mathtt{s}_4) = \mathsf{ap}(\mathtt{s}_7) = \emptyset$ *and* $\mathsf{ap}(\mathtt{s}_1) = \mathsf{ap}(\mathtt{s}_5) = \mathsf{ap}(\mathtt{s}_6) = \{p\}$. *The transition function is directly derivable from the figure.*

3.4.1 Normalization

In this subsection, we introduce the concept of normalized CGS *w.r.t.* a given binding. The aim is to show how to turn a CGS $\mathcal{G}$ in a new one $\mathcal{G}^\bullet$ in which all agents associated with the same variable are merged into a single player. Basically, by applying the normalization, we restrict our attention to the part of the structure that is effectively involved in the verification of the formula *w.r.t.* a binding $\flat$. From a technical point of view, the normalization consists of two steps. The first transforms the set of variables into the set of agents; this means that all bindings become identities of the kind (x, x). The second involves the transition function, which is augmented in order to associate decisions to the new agent (via the binding).

Construction 3.4.1 (CGS Normalization) *From a* CGS $\mathcal{G}{=}\langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_0\rangle$, *a binding prefix* $\flat \in \mathrm{Bn}(\mathrm{Ag})$ *and* $\mathrm{GSL}[1\mathrm{G}]$ *formula* $\varphi = \wp\flat\psi$, *we build the* normalized CGS $\mathcal{G}^\bullet \triangleq \langle \mathrm{AP}, \mathrm{Ag}^\bullet, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}^\bullet, \mathsf{ap}, s_I\rangle$ *as follows:*

- *the new agents in* $\mathrm{Ag}^\bullet \triangleq \mathsf{rng}(\flat)$ *are all variables bounded by* $\flat$, *where* $\mathsf{rng} : \mathrm{Bn} \to \mathrm{Ag}$, i.e., *it returns all agents bounded by* $\flat$.
- *the new transition function* $\mathsf{tr}^\bullet(\delta^\bullet)(s) \triangleq \mathsf{tr}(\delta^\bullet \circ \flat)(s)$ *simply maps a state* $s \in \mathrm{St}$ *and a new active decision* $\delta^\bullet \in \mathsf{dc}^\bullet(s) \triangleq \{\delta^\bullet \in \mathrm{Dc}^\bullet : \delta^\bullet \circ \flat \in \mathsf{dc}(s)\}$ *into the successor* $\mathsf{tr}(\delta^\bullet \circ \flat)(s)$ *of* s *following the original decision* $\delta^\bullet \circ \flat \in \mathrm{Dc}$;
- *the new* $\mathrm{GSL}[1\mathrm{G}]$ *formula is* $\varphi^\bullet \triangleq \wp \prod_{x \in \mathsf{rng}(\flat)}(x, x)\psi$.

Observe that, to normalize the game, we simply need to normalize its CGS, as well as to change the underling $\mathrm{GSL}[1\mathrm{G}]$ formula, since agent and variable names now coincide. Indeed, the new $\mathrm{GSL}[1\mathrm{G}]$ formula differs from the original one only on its bindings, which now are all identities.

Example 3.4.2 *Consider again the game depicted in Figure 3.2, with* $\varphi = \wp\flat\mathrm{F}p$ *a* $\mathrm{GSL}[1\mathrm{G}]$ *formula, where* $\wp = \langle\!\langle \mathtt{x} \geq 3 \rangle\!\rangle [\![\mathtt{y} < 2]\!]$ *and* $\flat = (\mathtt{a}, \mathtt{x})(\mathtt{b}, \mathtt{y})(\mathtt{c}, \mathtt{x})$. *The resulting normalized* CGS *is* $\mathcal{G}^\bullet \triangleq \langle \mathrm{AP}, \mathrm{Ag}^\bullet, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}^\bullet, \mathsf{ap}, s_I\rangle$, *where the set of new agents is* $\mathrm{Ag}^\bullet \triangleq \{\mathtt{x}, \mathtt{y}\}$ *and the transition function is reported in Figure 3.3. Note that, the transitions in which the agents* $\mathtt{a}$ *and* $\mathtt{c}$ *in Figure 3.2 take different actions are removed. The associated* $\mathrm{GSL}[1\mathrm{G}]$ *sentence is* $\varphi^\bullet \triangleq \langle\!\langle \mathtt{x} \geq 3 \rangle\!\rangle [\![\mathtt{y} < 2]\!](\mathtt{x}, \mathtt{x})(\mathtt{y}, \mathtt{y})\mathrm{F}p$.

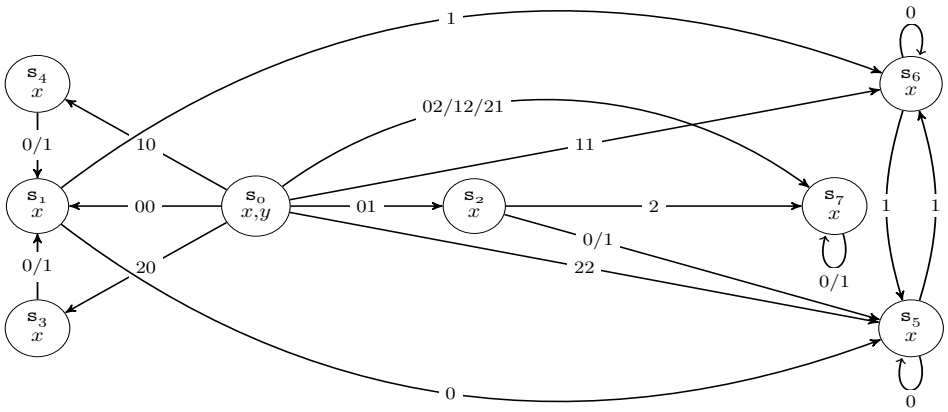


Figure 3.3: Normalized CGS $\mathcal{G}^\bullet$ built on $\mathcal{G}$.

3.4.2 Minimization

As for previous considerations, actions involving the same strategic reasoning need to be merged together. We accomplish this by constructing a new concurrent game structure that maintains just one representative for each class of equivalence actions. Before describing the formal construction, we need to introduce some accessory notions.

Given a CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_o \rangle$, one of its states $s \in \mathrm{St}$, a quantification prefix $\wp \in \mathrm{Qn}(\mathsf{ag}(s))$, and a function $\mathsf{vr} : \mathrm{Qn} \to \mathrm{Vr}$, we can define an equivalence relation $\delta_1 \equiv_s^{\wp} \delta_2$ between decisions $\delta_1, \delta_2 \in \mathrm{Dc}$ with $\mathsf{ag}(s) \setminus \mathsf{vr}(\wp) \subseteq \mathsf{dom}(\delta_1), \mathsf{dom}(\delta_2)$ that locally mimics the behavior of the one between assignments previously discussed. Intuitively, it allows to determine whether two different moves of a set of agents are actually mutually substitutable *w.r.t.* the strategy quantification of interest. Formally, we have that:

1. for the empty quantification prefix ϵ, it holds that $\delta_1 \equiv_s^{\epsilon} \delta_2$ iff $\mathsf{tr}(\delta_1)(s) = \mathsf{tr}(\delta_2)(s)$;
2. $\delta_1 \equiv_s^{\langle\!\langle a \geq g \rangle\!\rangle \wp} \delta_2$ iff, for all active actions $c \in \mathsf{ac}(s, a)$, it holds that $\delta_1[a \mapsto c] \equiv_s^{\wp} \delta_2[a \mapsto c]$;
3. $\delta_1 \equiv_s^{[\![a < g]\!] \wp} \delta_2$ iff, for all indexes $i \in \{1, 2\}$ and active actions $c_i \in \mathsf{ac}(s, a)$, there exists an active action $c_{3-i} \in \mathsf{ac}(s, a)$ such that $\delta_1[a \mapsto c_1] \equiv_s^{\wp} \delta_2[a \mapsto c_2]$.

At this point, we can introduce an equivalence relation between the active actions $c_1, c_2 \in \mathsf{ac}(s, a)$ of an agent $a \in \mathsf{ag}(s)$, once a partial decision $\delta \in \mathrm{Dc}$ with $\{a' \in \mathsf{ag}(s) : a' <_{\wp} a\} \subseteq \mathsf{dom}(\delta)$ of the agents already quantified is given. Formally, $c_1 \equiv_{s,\delta}^{\langle\!\langle a \geq g \rangle\!\rangle \wp} c_2$ iff $\delta[a \mapsto c_1] \equiv_s^{\wp} \delta[a \mapsto c_2]$ and $c_1 \equiv_{s,\delta}^{[\![a < g]\!] \wp} c_2$ iff $\delta[a \mapsto c_1] \equiv_s^{\overline{\wp}} \delta[a \mapsto c_2]$, where $\overline{\wp}$ represents the dual

prefix of $\wp$, *i.e.*, $\overline{\wp} = \neg\wp$. Intuitively, the two actions c_1, c_2 are equivalent *w.r.t.* δ iff agent a can use them indifferently to extend δ, without changing the set of successors of s it can force to reach.

We can now introduce the concept of minimization of a CGS, in which the behavior of each agent is restricted in such a way that he can only choose the representative element from each class of equivalent actions. Before moving to the formal definition, as an additional notation, we use $\wp_{\geq a}$ to denote the suffix of the quantification prefix $\wp$ starting from its variable/agent a.

Construction 3.4.2 (CGS **Minimization)** *From a* CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_0 \rangle$ *normalized* w.r.t. *a binding prefix* $\flat \in \mathrm{Bn}(\mathrm{Ag})$*, and a quantification prefix* $\wp \in \mathrm{Qn}(\mathsf{rng}(\flat))$*, we build the* minimized CGS $\mathcal{G}^{\blacklozenge} \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}^{\blacklozenge}, \mathsf{ap}, s_I \rangle$*, where the new transition function* $\mathsf{tr}^{\blacklozenge}$ *is defined as follows. First, assume* $\Lambda(s, \delta, a) \subseteq \mathsf{ac}(s, a)$ *to be a subset of active actions for the agent* $a \in \mathsf{ag}(s)$ *on the state* $s \in \mathrm{St}$ *such that, for each* $c \in \mathsf{ac}(s, a)$*, there is exactly one* $c' \in \Lambda(s, \delta, a)$ *with* $c \equiv_{s,\delta}^{\wp_{\geq a}} c'$*. Intuitively,* $\Lambda(s, \delta, a)$ *is one of the minimal sets of actions needed by the agent* a *in order to preserve the essential structure of the* CGS*. At this point, let* $\mathsf{dc}^{\blacklozenge}(s) \triangleq \{\delta \in \mathsf{dc}(s) : \forall a \in \mathsf{dom}(\delta) \,.\, \delta(a) \in \Lambda(s, \delta_{\restriction\{a' \in \mathsf{dom}(\delta) : a' <_{\wp} a\}}, a)\}$ *to be the set of active decisions having only values among those ones previously chosen. Finally, for each state* $s \in \mathrm{St}$ *and decision* $\delta \in \mathrm{Dc}$*, assume* $\mathsf{tr}^{\blacklozenge}(\delta)(s) \triangleq \mathsf{tr}(\delta)(s)$*, if* $\delta \in \mathsf{dc}^{\blacklozenge}(s)$*, and* $\mathsf{tr}^{\blacklozenge}(\delta) \triangleq \emptyset$*, otherwise.*

Observe that the minimization of the game only involves the CGS, as we just change the active actions of the agents, while states and agents remain unchanged.

Example 3.4.3 *Consider the normalized game* $\mathcal{G}^{\bullet}$ *of the Example 3.4.2 and sentence* $\varphi^{\bullet} = \wp\flat\mathtt{F}p$*, where* $\wp = \langle\langle \mathtt{x} \geq 3 \rangle\rangle [\![\mathtt{y} < 2]\!]$ *and* $\flat = (\mathtt{x}, \mathtt{x})(\mathtt{y}, \mathtt{y})$*. The corresponding minimized* CGS *is* $\mathcal{G}^{\blacklozenge} \triangleq \langle \mathrm{AP}, \mathrm{Ag}^{\bullet}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}^{\blacklozenge}, \mathsf{ap}, s_I \rangle$*, where the new transition function* $\mathsf{tr}^{\blacklozenge}$ *is depicted in Figure 3.4. To give an intuition, we analyze the equivalence relation between the actions* $0, 1 \in \mathsf{ac}(\mathtt{s}_3, \mathtt{x})$ *of the agent* x*. We have that* $0 \equiv_{\mathtt{s}_3, \varnothing}^{\wp} 1$ *iff* $\varnothing[\mathtt{x} \mapsto 0] \equiv_{\mathtt{s}_3}^{[\![\mathtt{y} < 2]\!]} \varnothing[\mathtt{x} \mapsto 1]$ *iff, for all indexes* $i \in \{1, 2\}$ *and active actions* $\mathtt{c}_i \in \mathsf{ac}(\mathtt{s}_3, \mathtt{y})$*, there exists an active action* $\mathtt{c}_{3-i} \in \mathsf{ac}(\mathtt{s}_3, \mathtt{y})$ *such that* $\delta_1[\mathtt{y} \mapsto \mathtt{c}_1] \equiv_{\mathtt{s}_3}^{\epsilon} \delta_2[\mathtt{y} \mapsto \mathtt{c}_2]$*. Since* $\mathsf{ac}(\mathtt{s}_3, \mathtt{y}) = \emptyset$ *the previous equivalence is vacuously verified. Therefore,* 0 *and* 1 *are equivalent actions.*

3.4.3 Conversion

Finally, we describe the conversion of concurrent game structures into turn-based ones. As anticipated before, differently from similar transformations one can found in literature, the game we obtain is one with k agents and k variables, where k is the number of variables of the starting game. Additionally, our construction makes use of the concepts of minimization and equivalence between actions, by removing the ones that induce equivalent paths. The

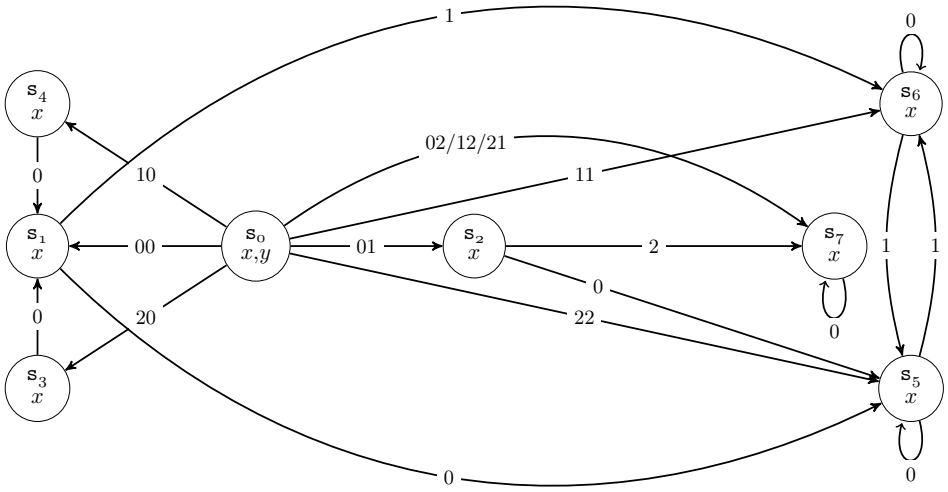


Figure 3.4: Minimized CGS $\mathcal{G}^{\blacklozenge}$ built on Normalized $\mathcal{G}^{\bullet}$.

intuitive idea of our reduction is to replace each state in the concurrent game structure with a finite tree whose height depends on the number of strategy quantifications. Also, we enrich each state of the new structure with extra information regarding the corresponding state in the concurrent one: *(i)* the index of the operator in the prefix of quantifications; *(ii)* the sequence of actions taken by the agents along a partial play. The formal definition follows.

Construction 3.4.3 (CGS **Conversion)** *From a* CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_{\mathrm{o}} \rangle$ *minimized* w.r.t. *a binding prefix* $\flat \in \mathrm{Bn}(\mathrm{Ag})$ *and a quantification prefix* $\wp \in \mathrm{Qn}(\mathsf{rng}(\flat))$*, we build the* TBGS $\mathcal{G}^{\star} \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}^{\star}, \mathsf{tr}^{\star}, \mathsf{ap}^{\star}, {s_I}^{\star} \rangle$*, where the new set of states* $\mathrm{St}^{\star}$ *and the new transition function* $\mathsf{tr}^{\star}$ *are defined as follows. Given a state* $s \in \mathrm{St}$*, we denote by* $\wp^s$ *the quantification prefix obtained from* $\wp$ *by simply deleting all agents/variables not in* $\mathsf{ag}(s)$ *and by* $\mathrm{Vr}(\wp^s)$ *the corresponding set of variables. The state space has to maintain the information about the position in* $\mathcal{G}$ *together with the index of the first variable that has still to be evaluated and the values already associated to the previous variables. To do this, we set* $\mathrm{St}^{\star} \triangleq \{(s, i, \delta) | s \in \mathrm{St}, i \in [0, |\mathsf{ag}(s)|], \delta \in (\mathrm{Vr}(\wp^s_{<i}) \to \mathrm{Ac})\}$*. Observe that, when a play is in a state* $(s, |\mathsf{ag}(s)|, \delta)$*, all quantifications are already resolved and it is time to evaluate the corresponding decision* δ*. Before proceeding with the definition of the transition function, it is helpful to identify which are the active agents and decisions for each possible state. Formally, for all* $(s, i, \delta) \in \mathrm{St}^{\star}$*, we have that* $\mathsf{ag}((s, i, \delta)) \triangleq \{\mathsf{vr}(\wp^s_i)\}$ *and* $\mathsf{dc}^{\star}((s, i, \delta)) \triangleq \{\mathsf{vr}(\wp^s_i) \mapsto c : c \in \mathsf{ac}(s, \mathsf{vr}(\wp^s_i))\}$*, if* $i < |\wp^s|$*, and* $\mathsf{ag}((s, i, \delta)) \triangleq \emptyset$ *and* $\mathsf{dc}^{\star}((s, i, \delta)) \triangleq \{\varnothing\}$*, otherwise. The transition function is de-*

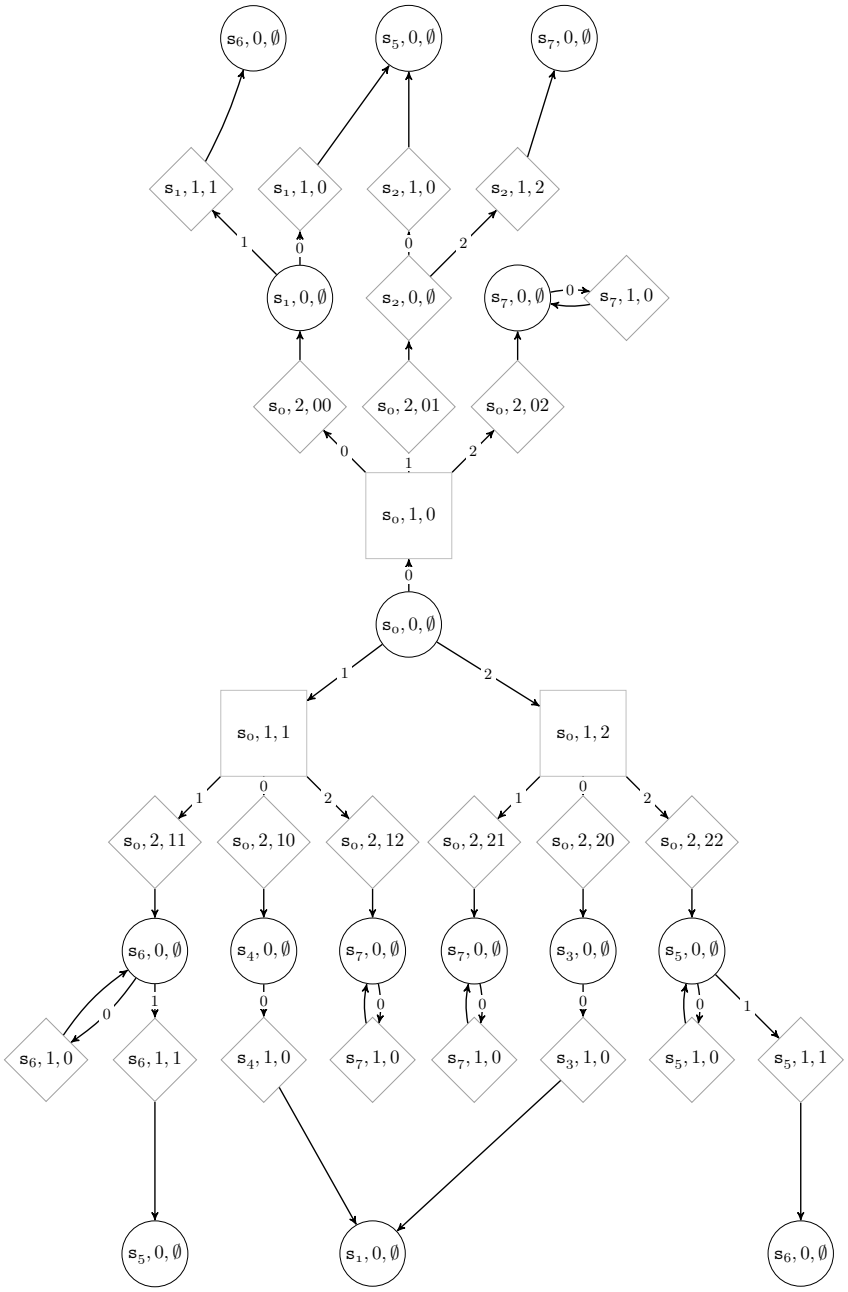


Figure 3.5: Turn-based game structure $\mathcal{G}^\star$ built on Minimized $\mathcal{G}^\blacklozenge$. In particular, the agent x is owner of all circle nodes, the agent y is owner of all square nodes, and each diamond node represents the transition state. Note that, for a matter of readability some nodes are duplicated.

fined as follows. For each new state (s, i, δ) *with* $i < |\wp^s|$ *and new decision* $\mathsf{vr}(\wp_i^s) \mapsto c$*, we simply need to increase the counter* i *and embed* $\mathsf{vr}(\wp_i^s) \mapsto c$ *into* δ*. Formally, we set* $\mathsf{tr}^\star(\mathsf{vr}(\wp_i^s) \mapsto c)((s, i, \delta)) \triangleq (s, i+1, \delta[\mathsf{vr}(\wp_i^s) \mapsto c])$*. For a new state* $(s, |\mathsf{ag}(s)|, \delta)$*, instead, we just introduce a transition to the state* $(s', 0, \varnothing)$*, where* s' *is the successor of* s *in the* CGS *following the decision* δ*. Formally, we have* $\mathsf{tr}^\star(\varnothing)((s, |\mathsf{ag}(s)|, \delta)) \triangleq (\mathsf{tr}(\delta)(s), 0, \varnothing)$*. The new labeling function* $\mathsf{ap}^\star$ *is such that, for each state* (s, j, δ) *we have that*

$$\mathsf{ap}^\star((s, j, \delta)) \triangleq \begin{cases} \mathsf{ap}(s), & \text{if } j = 0 \text{ and } \delta = \emptyset; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Finally, the initial state $s_I^\star \triangleq (s_I, 0, \varnothing)$*.*

By means of a simple generalization of the classic correctness proof of a transformation of a concurrent game into a turn-based one, the following result derives.

Theorem 3.4.1 (Concurrent/Turn-Based Conversion) *For each* CGS $\mathcal{G}$ *with* $|\mathrm{St}|$ *and* $\mathrm{GSL}[1\mathrm{G}]$ *formula* $\varphi = \wp\flat\psi$ *with* $|\mathrm{Vr}(\wp)|$ *variables, there is an equivalent* TBGS $\mathcal{G}^\star$ *with* $|\mathrm{Vr}(\wp)|$ *agents/variables of order* $\mathrm{O}(|\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathrm{Vr}(\wp)|})$*.*

Proof. The theorem is proved by following the three steps of *normalization*, *minimization* and *conversion*. In detail, from a CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_0 \rangle$, a binding prefix $\flat \in \mathrm{Bn}(\mathrm{Ag})$ and $\mathrm{GSL}[1\mathrm{G}]$ formula $\varphi = \wp\flat\psi$, by applying the construction described in Sections 3.4.1, we obtain the *normalized* CGS $\mathcal{G}^\bullet \triangleq \langle \mathrm{AP}, \mathrm{Ag}^\bullet, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}^\bullet, \mathsf{ap}, s_I \rangle$. At this point, From the latter *w.r.t.* a binding prefix $\flat \in \mathrm{Bn}(\mathrm{Ag})$, and a quantification prefix $\wp \in \mathrm{Qn}(\mathsf{rng}(\flat))$, by applying the construction described in Sections 3.4.2 we build the *minimized* CGS $\mathcal{G}^\blacklozenge \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}^\blacklozenge, \mathsf{ap}, s_I \rangle$. Finally, , we build the TBGS $\mathcal{G}^\star \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}^\star, \mathsf{tr}^\star, \mathsf{ap}^\star, s_I{}^\star \rangle$ by applying the construction in Sections 3.4.3 to the minimized CGS $\mathcal{G}^\blacklozenge \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}^\blacklozenge, \mathsf{ap}, s_I \rangle$. Regarding the complexity of the conversion from $\mathcal{G}$ to $\mathcal{G}^\star$, we have that the size of $\mathcal{G}^\star$ is exponential in the number of the variable of the quantification prefix. Indeed, for each $s \in \mathrm{St}$, the conversion produces a number of states equal to $\sum_{i=0}^{|\mathsf{ag}(s)|} |\mathrm{Ac}|^i = \mathrm{O}(|\mathrm{Ac}|^{|\mathsf{ag}(s)|})$. So, the overall size is $\mathrm{O}(|\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathsf{ag}(s)|})$. Thanks to the normalization $\mathrm{Ag} = \mathrm{Vr}(\wp)$, the result follows. □

Example 3.4.4 *Consider the minimized game* $\mathcal{G}^\blacklozenge$ *of Example 3.4.3 with the formula* $\varphi^\blacklozenge = \varphi^\bullet$*. We want to build a turn-based game* $\mathcal{G}^\star$ *from* $\mathcal{G}^\blacklozenge$*. The new* CGS *is* $\mathcal{G}^\star \triangleq \langle \mathrm{AP}, \mathrm{Ag}^\bullet, \mathrm{Ac}, \mathrm{St}^\star, \mathsf{tr}^\star, \mathsf{ap}^\star, s_I{}^\star \rangle$*, where the new set of states* $\mathrm{St}^\star$*, the new transition function* $\mathsf{tr}^\star$*, and the new initial state* $s_I{}^\star \triangleq (s_0, 0, \emptyset)$ *are depicted in Figure 3.5. Finally, the labeling function is* $\mathsf{ap}(\mathsf{s}_0, 0, \emptyset) = \mathsf{ap}(\mathsf{s}_2, 0, \emptyset) = \mathsf{ap}(\mathsf{s}_3, 0, \emptyset) = \mathsf{ap}(\mathsf{s}_4, 0, \emptyset) = \mathsf{ap}(\mathsf{s}_7, 0, \emptyset) = p$ *and, for each state* (s, j, δ)*, with* $j \neq 0$ *and* $\delta \neq \emptyset$ *we have that* $\mathsf{ap}(s, j, \delta) = \emptyset$*.*

3.5 Determinacy

In this section, we address the determinacy problem for a fragment of GSL, that we name GSL[1G, 2AG], involving only two players over turn-based structures. Recall that determinacy has been first proved for classic Borel turn-based two-player games in [Mar75]. However, the proof used there does not directly apply to our graded setting. To give evidence of the differences between the two frameworks, observe that in SL[1G, 2AG] sentences like $\langle\langle x\rangle\rangle[\![\overline{x}]\!]\eta$ imply $[\![\overline{x}]\!]\langle\langle x\rangle\rangle\eta$, while in GSL[1G, 2AG] the corresponding implication $\langle\langle x \geq i\rangle\rangle[\![\overline{x} < j]\!]\eta \Rightarrow [\![\overline{x} < j]\!]\langle\langle x \geq i\rangle\rangle\eta$ does not hold. The determinacy property we are interested in is exactly the converse direction, *i.e.*, $[\![\overline{x} < j]\!]\langle\langle x \geq i\rangle\rangle\eta \Rightarrow \langle\langle x \geq i\rangle\rangle[\![\overline{x} < j]\!]\eta$. In particular, we extend the Gale-Stewart Theorem [PP04], by exploiting a deep generalization of the technique used in [FNP09a]. The idea consists of a fixed-point calculation over the number of winning strategies an agent can select against all but a fixed number of those of its opponent. Regarding this approach, we recall that the simpler counting considered in [FNP09a] is restricted to existential quantifications only.

Construction 3.5.1 (Grading Function) *Let $\mathcal{G}$ be a two-agent turn-based game structure $\mathcal{G}$ with* $\mathrm{Ag} = \{\alpha, \overline{\alpha}\}$*, and ψ be an* LTL *formula with* $\mathrm{W}_{\psi}, \mathrm{W}_{\neg\psi} \subseteq \mathrm{Hst}$ *denoting the witness sets for ψ and $\neg\psi$, respectively. It is immediate to see that, in case $s_I \in \mathrm{W}_{\psi}$ (resp., $s_I \in \mathrm{W}_{\neg\psi}$), all strategy profiles are equivalent* w.r.t. *the temporal property ψ (resp., $\neg\psi$). If $s_I \in \mathrm{X} \triangleq \mathrm{Hst} \setminus (\mathrm{W}_{\psi} \cup \mathrm{W}_{\neg\psi})$, instead, we need to introduce a* grading function $\mathsf{G}^{\alpha}_{\psi} : \mathrm{X} \to \Gamma$*, where $\Gamma \triangleq \mathbb{N} \to (\mathbb{N} \cup \{\omega\})$, that allows to determine how many different strategies the agent α (resp., $\overline{\alpha}$) owns* w.r.t. *ψ (resp., $\neg\psi$). Informally, $\mathsf{G}^{\alpha}_{\psi}(\rho)(j)$ represents the number of winning strategies player α can put up against all but at most j strategies of its adversary $\overline{\alpha}$, once the current play has already reached the history $\rho \in \mathrm{X}$.*

Before continuing, observe that α sometimes has the possibility to commit a suicide, i.e.*, to choose a strategy leading directly to a history in $\mathrm{W}_{\neg\psi}$, with the hope to win the game by collapsing all strategies of its opponent into a unique class. The set of histories enabling this possibility is defined as follows:* $\mathrm{S} \triangleq \{\rho \in \mathrm{X} : \exists \rho' \in \mathrm{W}_{\neg\psi} \,.\, \rho < \rho' \wedge \forall \rho'' \in \mathrm{Hst} \,.\, \rho \leq \rho'' < \rho' \Rightarrow \rho'' \in \mathrm{Hst}_{\alpha}\}$*, where* $\mathrm{Hst}_{\alpha} = \{\rho \in \mathrm{Hst} : \mathsf{ag}(\mathsf{lst}(\rho)) = \{\alpha\}\}$ *is the set of histories ending in a state controlled by α. Intuitively, α can autonomously extend a history $\rho \in \mathrm{S}$ into one $\rho' \in \mathrm{W}_{\neg\psi}$ that is surely loosing, independently of the behavior of $\overline{\alpha}$. Note that there may be several suicide strategies, but all of them are equivalent* w.r.t. *the property ψ. Also, against them, all counter strategies of $\overline{\alpha}$ are equivalent as well.*

At this point, to define the function $\mathsf{G}^{\alpha}_{\psi}$, we introduce the auxiliary functor $\mathsf{F}^{\alpha}_{\psi} : (\mathrm{X} \to \Gamma) \to (\mathrm{X} \to \Gamma)$, whose least fixpoint represents a function returning the maximum number of different strategies α can use against all but a precise fixed number of counter strategies of $\overline{\alpha}$. Formally, we have that:

$$\mathsf{F}^{\alpha}_{\psi}(\mathsf{f})(\rho)(j) \triangleq \begin{cases} \sum_{\rho' \in \mathsf{suc}(\rho) \cap \mathrm{X}} \mathsf{f}(\rho')(0) + |\mathsf{suc}(\rho) \cap \mathrm{W}_{\psi}|, & \text{if } \rho \in \mathrm{Hst}_{\alpha} \text{ and } j = 0; \\ \sum_{\rho' \in \mathsf{suc}(\rho) \cap \mathrm{X}} \mathsf{f}(\rho')(j), & \text{if } \rho \in \mathrm{Hst}_{\alpha} \text{ and } j > 0; \\ \sum_{\mathsf{c} \in \mathrm{C}(\rho)(j)} \prod_{\rho' \in \mathsf{dom}(\mathsf{c})} \mathsf{f}(\rho')(\mathsf{c}(\rho')), & \text{otherwise}; \end{cases}$$

where $\mathsf{suc}(\rho) = \{\rho' \in \mathrm{Hst} : \exists s \in \mathrm{St}.\ \rho s = \rho'\}$ *and* $\mathsf{C}(\rho)(i) \subseteq (\mathsf{suc}(\rho) \cap \mathrm{X}) \rightharpoonup \mathbb{N}$ *contains all partial functions* $\mathsf{c} \in \mathsf{C}(\rho)(i)$ *for which* α *owns a suicide strategy on the histories not in their domains,* i.e., $(\mathsf{suc}(\rho) \cap \mathrm{X}) \setminus \mathsf{dom}(\mathsf{c}) \subseteq \mathrm{S}$*, and the sum of all values assumed by* c *plus the number of successor histories that are neither surely winning nor contained in the domain of* c *equals to* i*,* i.e., $i = \sum_{\rho' \in \mathsf{dom}(\mathsf{c})} \mathsf{c}(\rho') + |\mathsf{suc}(\rho) \setminus (\mathrm{W}_{\psi} \cup \mathsf{dom}(\mathsf{c}))|$.

Intuitively, the first item of the definition simply asserts that the number of strategies $\mathsf{F}(\mathsf{f})(\rho)(0)$ *that agent* α *has on the* α*-history* ρ*, without excluding any counter strategy of its adversary, is obtainable as the sum of the* $\mathsf{f}(\rho')(0)$ *strategies on the successor histories* $\rho' \in \mathrm{X}$ *plus a single strategy for each successor history that is surely winning. Similarly, the second item takes into account the case in which we can avoid exactly* j *counter strategies. The last item, instead, computes the number of strategies for* α *on the* $\overline{\alpha}$*-histories. In particular, through the set* $\mathsf{C}(\rho)(j)$*, it first determines in how many ways it is possible to split the number* j *of counter strategies to avoid among all successor histories of* ρ*. Then, for each of these splittings, it calculates the product of the corresponding numbers* $\mathsf{f}(\rho')(\mathsf{c}(\rho'))$ *of strategies for* α*.*

We are finally able to define the grading function $\mathsf{G}^{\alpha}_{\psi}$ *by means of the least fixpoint* $\mathsf{f}^{\star} = \mathsf{F}^{\alpha}_{\psi}(\mathsf{f}^{\star})$ *of the functor* $\mathsf{F}^{\alpha}_{\psi}$*, whose existence is proved in Lemma 3.5.1:*

$$\mathsf{G}^{\alpha}_{\psi}(\rho)(j) \triangleq \sum_{h=0}^{j} \mathsf{f}^{\star}(\rho)(h) + \begin{cases} 1, & \text{if } \rho \in \mathrm{S} \text{ and } j \geq 1; \\ 0, & \text{otherwise.} \end{cases}$$

Intuitively, $\mathsf{G}^{\alpha}_{\psi}(\rho)(j)$ *is the sum of the numbers* $\mathsf{f}^{\star}(\rho)(h)$ *of winning strategies the agent* α *can exploit against all but exactly* h *strategies of its adversary* $\overline{\alpha}$*, for each* $h \in [0, j]$*. Moreover, if* $\rho \in \mathrm{S}$*, we need to add to this counting the suicide strategy that* α *can use once* $\overline{\alpha}$ *avoids to apply his unique counter strategy.*

Lemma 3.5.1 (Fixpoint Existence) *The functor* $\mathsf{F}^{\alpha}_{\psi}$ *of Construction 3.5.1 admits a unique least-fixed point.*

Proof. Consider the set of functions $\mathrm{D} \triangleq \mathrm{X} \to \Gamma$, where $\Gamma \triangleq \mathbb{N} \to (\mathbb{N} \cup \{\omega\})$, equipped with the binary relation $\sqsubseteq \subseteq \mathrm{D} \times \mathrm{D}$ defined as follows: $\mathsf{f}_1 \sqsubseteq \mathsf{f}_2$ iff $\mathsf{f}_1(\rho)(j) \leq \mathsf{f}_2(\rho)(j)$, for all histories $\rho \in \mathrm{X}$ and indexes $j \in \mathbb{N}$, where $\leq$ is the standard ordering on the set of natural numbers extended with the maximum element ω. Now, it is immediate to see that $\sqsubseteq$ is a reflexive, antisymmetric, and transitive relation over D. Hence, $(\mathrm{D}, \sqsubseteq)$ is a partial

order. Actually, this structure is a complete lattice [Win93], since any set of functions $\mathrm{F} \subseteq \mathrm{D}$ admits a greatest lower bound $\inf \mathrm{F}$, which can be computed as follows: $(\inf \mathrm{F})(\rho)(j) \triangleq \min_{\mathsf{f} \in \mathrm{F}} \mathsf{f}(\rho)(j)$, for all $\rho \in \mathrm{X}$ and $j \in \mathbb{N}$. Moreover, by direct inspection, it can be easily showed that the functor $\mathsf{F}^{\alpha}_{\psi} : \mathrm{D} \to \mathrm{D}$ over D defined in Construction 3.5.1 is monotone *w.r.t.* $\sqsubseteq$, *i.e.*, $\mathsf{F}^{\alpha}_{\psi}(\mathsf{f}_1) \sqsubseteq \mathsf{F}^{\alpha}_{\psi}(\mathsf{f}_2)$, whenever $\mathsf{f}_1 \sqsubseteq \mathsf{f}_2$, for all $\mathsf{f}_1, \mathsf{f}_2 \in \mathrm{D}$ (in particular, notice that the only operations used in its definition are the sum and the multiplication). Consequently, by the Knaster-Tarski Theorem, $\mathsf{F}^{\alpha}_{\psi}$ admits a least fixpoint. □

Thanks to the above construction, one can compute the maximum number of strategies that a player has at its disposal against all but a fixed number of strategies of the opponent. Next lemma precisely describes this fact. Indeed, we show how the satisfiability of a $\mathrm{GSL}[1\mathrm{G}, 2\mathrm{AG}]$ sentence $\langle\!\langle x \geq i \rangle\!\rangle [\![\overline{x} \leq j]\!](\alpha, x)(\overline{\alpha}, \overline{x})\psi$ can be decided via the computation of the associated grading function $\mathsf{G}^{\alpha}_{\psi}$, where by $[\![\overline{x} \leq j]\!]\varphi$ we mean $[\![\overline{x} < j + 1]\!]\varphi$.

Lemma 3.5.2 (Grading Function) *Let $\mathcal{G}$ be a two-agent turn-based game structure, where $\mathrm{Ag} = \{\alpha, \overline{\alpha}\}$, and $\varphi = \langle\!\langle x \geq i \rangle\!\rangle [\![\overline{x} \leq j]\!](\alpha, x)(\overline{\alpha}, \overline{x})\psi$ a $\mathrm{GSL}[1\mathrm{G}, 2\mathrm{AG}]$ sentence. Moreover, let $\mathsf{G}^{\alpha}_{\psi}$ be the grading function and $\mathrm{W}_{\psi}, \mathrm{W}_{\neg\psi}, \mathrm{X} \subseteq \mathrm{Hst}$ the sets of histories obtained in Construction 3.5.1. Then, $\mathcal{G} \models \varphi$ iff one of the following three conditions hold:* (i) $i \leq 1$, $j \geq 0$, *and* $s_I \in \mathrm{W}_{\psi}$; (ii) $i \leq 1$, $j \geq 1$, *and* $s_I \in \mathrm{W}_{\neg\psi}$; (iii) $i \leq \mathsf{G}^{\alpha}_{\psi}(s_I)(j)$ *and* $s_I \in \mathrm{X}$.

Proof. For the case *(i)*, we consider the worst scenario in which $i = 1$ and $j = 0$, *i.e.*, we have the sentence $\varphi = \langle\!\langle x \geq 1 \rangle\!\rangle [\![\overline{x} \leq 0]\!](\alpha, x)(\overline{\alpha}, \overline{x})\psi$. Since $s_I \in \mathrm{W}_{\psi}$ and W_{ψ} only contains histories that cannot be extended to a play violating the property ψ, we know that from s_I, by taking any strategy for player α against all strategies for the player $\overline{\alpha}$, the corresponding play satisfy the formula ψ. Moreover, all strategies are equivalent. We show this by directly analyzing the semantic of sentence φ.

1. $\mathcal{G} \models \varphi$ iff $|(\{\varnothing[x \mapsto \sigma_x] : \sigma_x \in \varphi'[\mathcal{G}, \varnothing, s_I](x)\}/\equiv^{\varphi'}_{\mathcal{G}})| \geq 1$, where $\varphi' = [\![\overline{x} \leq 0]\!](\alpha, x)(\overline{\alpha}, \overline{x})\psi$;

2. $\varphi'[\mathcal{G}, \varnothing, s_I](x) = \{\sigma_x \in \mathrm{Str}(\{\alpha\}) : \mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I \models \varphi'\}$;

3. $\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I \models \varphi'$ iff $|(\{\varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}] : \sigma_{\overline{x}} \in \neg\varphi''[\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x})\}/\equiv^{\neg\varphi''}_{\mathcal{G}})| \leq 0$, where $\varphi'' = (\alpha, x)(\overline{\alpha}, \overline{x})\psi$;

4. $\neg\varphi''[\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x}) = \{\sigma_{\overline{x}} \in \mathrm{Str}(\{\overline{\alpha}\}) : \mathcal{G}, \varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}], s_I \models \neg\varphi''\}$;

5. $\mathcal{G}, \varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}], s_I \not\models \neg\varphi''$, since $\mathcal{G}, \varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}], s_I \models \varphi''$ due to the fact that $s_I \in \mathrm{W}_{\psi}$.

By item *(5)*, the set $\neg\varphi''[\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x})$ of item *(4)* is empty. Therefore, from the item *(3)* we immediately derive that $\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I \models \varphi'$. Consequently, the set $\varphi'[\mathcal{G}, \varnothing, s_I](x)$ of item *(2)* is equal to $\mathrm{Str}(\{\alpha\})$, from which we immediately see at item *(1)* that $\mathcal{G}, \varnothing, s_I \models \varphi$.

For the case *(ii)*, we consider the worst scenario in which $i = 1$ and $j = 1$, *i.e.* we have the sentence $\varphi = \langle\!\langle x \geq 1 \rangle\!\rangle [\![\overline{x} \leq 1]\!](\alpha, x)(\overline{\alpha}, \overline{x})\psi$. Since $s_I \in \mathrm{W}_{\neg\psi}$ then the player $\overline{\alpha}$ can use all its strategies to satisfy $\neg\psi$, independently from behavior of α. By Definition 3.3.3, we know that all these strategies are equivalent. Therefore, by removing the unique corresponding equivalence class we have that agent α can use any of its strategies to vacuously satisfy the formula ψ, since we do not actually require φ to hold at all. Also in this case, we show this by directly analyzing the semantic of sentence φ.

1. $\mathcal{G} \models \varphi$ iff $|(\{\varnothing[x \mapsto \sigma_x] : \sigma_x \in \varphi'[\mathcal{G}, \varnothing, s_I](x)\}/\equiv^{\varphi'}_{\mathcal{G}})| \geq 1$, where $\varphi' = [\![\overline{x} \leq 0]\!](\alpha, x)(\overline{\alpha}, \overline{x})\psi$;

2. $\varphi'[\mathcal{G}, \varnothing, s_I](x) = \{\sigma_x \in \mathrm{Str}(\{\alpha\}) : \mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I \models \varphi'\}$;

3. $\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I \models \varphi'$ iff $|(\{\varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}] : \sigma_{\overline{x}} \in \neg\varphi''[\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x})\}/\equiv^{\neg\varphi''}_{\mathcal{G}})| \leq 1$, where $\varphi'' = (\alpha, x)(\overline{\alpha}, \overline{x})\psi$;

4. $\neg\varphi''[\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x}) = \{\sigma_{\overline{x}} \in \mathrm{Str}(\{\overline{\alpha}\}) : \mathcal{G}, \varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}], s_I \models \neg\varphi''\}$;

5. $\mathcal{G}, \varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}], s_I \models \neg\varphi''$, since $s_I \in \mathrm{W}_{\neg\psi}$.

By item *(5)*, the set $\neg\varphi''[\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x})$ of item *(4)* is equal to $\mathrm{Str}(\{\overline{\alpha}\})$. By Definition 3.3.3, for all $\sigma_1, \sigma_2 \in \neg\varphi''[\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x})$, it holds that $\varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_1] \equiv^{\neg\varphi''}_{\mathcal{G}} \varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_2]$. Due to this fact, $|(\{\varnothing[x \mapsto \sigma_x, \overline{x} \mapsto \sigma_{\overline{x}}] : \sigma_{\overline{x}} \in \neg\varphi''\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I](\overline{x})\}/\equiv^{\neg\varphi''}_{\mathcal{G}})| = 1$ we immediately derive that $\mathcal{G}, \varnothing[x \mapsto \sigma_x], s_I \models \varphi'$. Consequently, the set $\varphi'[\mathcal{G}, \varnothing, s_I](x)$ of item *(2)* is equal to $\mathrm{Str}(\{\alpha\})$, from which we immediately see at item *(1)* that $\mathcal{G}, \varnothing, s_I \models \varphi$.

For the case *(iii)*, the proof proceeds by *nested induction* over the indexes j and i of strategy counting. Precisely, the external induction is done over j, while the internal one over i.

As internal base case, *i.e.*, when $j = 0$ and $i = 1$, we have that $\mathsf{G}^{\alpha}_{\psi}(s_I)(0) \triangleq \mathsf{f}^{\star}(s_I)(0) \geq 1$, where $\mathsf{f}^{\star}$ is the least fix point of the functor $\mathsf{F}^{\alpha}_{\psi}$, *i.e.*, $\mathsf{f}^{\star} = \mathsf{F}^{\alpha}_{\psi}(\mathsf{f}^{\star})$. Now, let $\sigma_\alpha \in \mathrm{Str}(\{\alpha\})$ be an α-strategy satisfying the following property: for all histories $\rho \in \mathrm{X} \cap \mathrm{Hst}_\alpha$ with $\mathsf{f}^{\star}(\rho)(0) \geq 1$, if $\mathsf{suc}(\rho) \cap \mathrm{W}_\psi \neq \emptyset$, then the action $\sigma_\alpha(\rho)$ is chosen in such a way that the successor history $\rho' \triangleq \rho \cdot \mathsf{tr}(\{\alpha \mapsto \sigma_\alpha(\rho)\})(\mathsf{lst}(\rho))$ of ρ following the decision $\{\alpha \mapsto \sigma_\alpha(\rho)\}$ belongs to $\mathsf{suc}(\rho) \cap \mathrm{W}_\psi$, *i.e.*, $\rho' \in \mathsf{suc}(\rho) \cap \mathrm{W}_\psi$. Otherwise, we require that $\mathsf{f}^{\star}(\rho')(0) \geq 1$. The existence of such a strategy is immediately derived by the first case of the definition of the functor $\mathsf{F}^{\alpha}_{\psi}$. Moreover, σ_α is a winning strategy for α, due to the last case of the same definition. To see that this is actually the case, let $\sigma_{\overline{\alpha}} \in \mathrm{Str}(\{\overline{\alpha}\})$ be an $\overline{\alpha}$-strategy and consider the resulting play $\pi = \mathsf{play}(\{\alpha \mapsto \sigma_\alpha, \overline{\alpha} \mapsto \sigma_{\overline{\alpha}}\}, s_I)$. Due to the construction of σ_α, on every history $\rho \in \mathrm{Hst}_\alpha$ that is a prefix of π, we have that $\mathsf{f}^{\star}(\rho)(0) \geq 1$. The same holds for every $\rho \in \mathrm{Hst}_{\overline{\alpha}}$ that is a prefix of π, as well. Indeed, due to the last case of the definition

of the functor, we would have had $\mathsf{f}^\star(\rho')(0) = 0$ otherwise, for all histories $\rho' \leq \rho$. However, this is clearly impossible, due to the fact that $\mathsf{f}^\star(s_I)(0) \geq 1$. Now, since $\mathsf{f}^\star$ is the least fix point of F^α_ψ, there exists necessarily a prefix $\rho \in \mathrm{Hst}$ of π belonging to W_ψ, which implies that π satisfies the temporal property ψ.

For the internal inductive case, *i.e.*, when $j = 0$ and $i > 1$, assume S_α to be the set of $i-1$ non-equivalent winning α-strategies constructed by inductive hypothesis. We want to prove that there exists a new α-strategy $\sigma_\alpha \in \mathrm{Str}(\{\alpha\})$ that is neither contained in S_α nor equivalent to any of those strategies there contained. To do this, let σ_α be the strategy satisfying the following property: for all histories $\rho \in \mathrm{X} \cap \mathrm{Hst}_\alpha$ with $\rho' \triangleq \rho \cdot \mathsf{tr}(\{\alpha \mapsto \sigma_\alpha(\rho)\})(\mathsf{lst}(\rho))$, it holds that $|\{\sigma \in \mathrm{S}_\alpha : \rho' = \rho \cdot \mathsf{tr}(\{\alpha \mapsto \sigma(\rho)\})(\mathsf{lst}(\rho))\}| < \mathsf{f}^\star(\rho')(0)$. Intuitively, the actions prescribed by σ_α force a play to follows histories that are not completely covered by the other strategies. Therefore, if such a strategy σ_α exists, we necessarily have that $\sigma_\alpha \notin \mathrm{S}_\alpha$. Moreover, due to the turn-based structure of the underlying model, this observation also suffices to prove that σ_α cannot be equivalent to any strategy contained in S_α. Indeed, due to the particular choice of the actions $\sigma_\alpha(\rho)$, there exists a play compatible with σ_α that is not compatible with any other strategy of this predetermined set. If, instead, such a particular strategy does not exist, there is a history $\rho \in \mathrm{X} \cap \mathrm{Hst}_{\overline{\alpha}}$ ruled by the opponent player $\overline{\alpha}$ satisfying the following: *(i)* for all prefixes $\rho' < \rho$, it holds that $\rho' \in \mathrm{Hst}_\alpha$; *(ii)* $|\{\sigma \in \mathrm{S}_\alpha : \forall \rho' < \rho \,.\, \rho' \cdot \mathsf{tr}(\{\alpha \mapsto \sigma(\rho')\})(\mathsf{lst}(\rho')) \leq \rho\}| < \mathsf{f}^\star(\rho)(0)$. Intuitively, these two properties ensure that the number of strategies of S_α passing trough ρ is strictly less than the one predicted by the function $\mathsf{f}^\star$. Consequently, there is an α-strategy $\sigma_\alpha \notin \mathrm{S}_\alpha$ such that $\rho' \cdot \mathsf{tr}(\{\alpha \mapsto \sigma_\alpha(\rho')\})(\mathsf{lst}(\rho')) \leq \rho$, for all $\rho' < \rho$. Also in this case σ_α is not equivalent to any strategy in S_α. Indeed, due to property *(ii)*, there always exists an $\overline{\alpha}$-strategy that forces σ_α and $\sigma \in \mathrm{S}_\alpha$ to follows different and, so, non equivalent plays. To conclude this case, one has to prove that σ_α is winning. To do this, the same approach used in the base case above can be applied.

Finally, for the remaining two cases having $j > 0$, we proceed similarly to the previous ones, by taking additional care to eliminate j strategies of player $\overline{\alpha}$ while proving that the considered i strategies of player α are winning. This is done by exploiting the splitting of all the $\overline{\alpha}$-strategies dictated by the set $\mathrm{C}(\rho)(j)$ used in the last case of the definition of the functor F^α_ψ. □

By transfinite induction on its recursive structure, we can prove a quite natural but fundamental property of the grading function, *i.e.*, its duality in the form described in the next lemma. To give an intuition, assume that agent $\overline{\alpha}$ has at most j strategies to satisfy the temporal property $\neg\psi$ against all but at most i strategies of its adversary α. Then, it can be shown that the latter has more than i strategies to satisfy ψ against all but at most j strategies of the former.

Lemma 3.5.3 (Grading Duality) *Let* G^α_ψ *and* $\mathsf{G}^{\overline{\alpha}}_{\neg\psi}$ *be the grading functions and* $\mathrm{X} \subseteq \mathrm{Hst}$

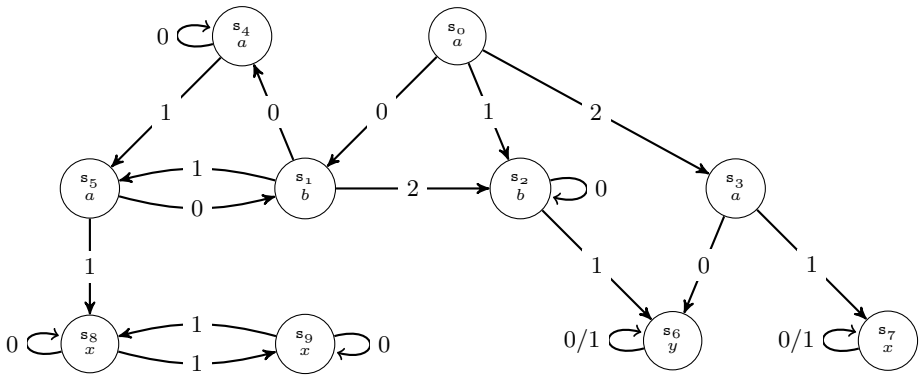


Figure 3.6: Turn-based structure.

the set of histories obtainable by Construction 3.5.1. For all histories $\rho \in \mathrm{X}$ *and indexes* $i, j \in \mathbb{N}$*, it holds that if* $\mathsf{G}^{\overline{\alpha}}_{\neg\psi}(\rho)(i) \leq j$ *then* $i < \mathsf{G}^{\alpha}_{\psi}(\rho)(j)$.

Summing up the above two results, we can easily prove that, on turn-based game structures, GSL[1G, 2AG] is determined. Indeed, suppose that $s_I \in \mathrm{X}$ and $\mathcal{G} \models [\![\overline{x} \leq j]\!]\langle\!\langle x \geq i\rangle\!\rangle \flat\psi$, where $\flat = (\alpha, x)(\overline{\alpha}, \overline{x})$ (the case with $s_I \in \mathrm{W}_\psi$ immediately follows from classic Martin's Determinacy Theorem [Mar75, Mar85]). Obviously, $\mathcal{G}$ does not satisfy the negation of this sentence, *i.e.*, $\mathcal{G} \not\models \langle\!\langle \overline{x} \geq j+1\rangle\!\rangle [\![x \leq i-1]\!] \flat\neg\psi$. By Lemma 3.5.2, we have that $\mathsf{G}^{\overline{\alpha}}_{\neg\psi}(s_I)(i-1) \leq j$. Hence, by Lemma 3.5.3, it follows that $i \leq \mathsf{G}^{\alpha}_{\psi}(s_I)(j)$. Finally, again by Lemma 3.5.2, we obtain that $\mathcal{G} \models \langle\!\langle x \geq i\rangle\!\rangle [\![\overline{x} \leq j]\!] \flat\psi$, as required by the definition of determinacy.

Theorem 3.5.1 (Determinacy) GSL[1G, 2AG] *on turn-based game structures is determined.*

Example 3.5.1 *Consider the structure depicted in Figure 3.6, the state* $\mathtt{s}_0 \in \mathrm{St}$*, and the formula* $\varphi = \langle\!\langle x \leq g_1\rangle\!\rangle [\![y < g_2]\!] \flat\psi$*, with* $\flat = (a, x)(b, y)$ *and* $\psi = \mathtt{F}p$*. The set of histories* W_ψ *is* $s_0 \cdot s_3 \cdot s_7^+ + s_0 \cdot (s_1 \cdot s_5)^+ \cdot s_8^+ \cdot s_9 \cdot (s_8 + s_9)^* + s_0 \cdot s_1 \cdot ((s_5 \cdot s_1)^* \cdot s_4)^+ \cdot (\epsilon + s_5 + s_5 \cdot (s_1 + s_8 \cdot (s_8 + s_9)^*))$*, while* $\mathrm{W}_{\neg\psi} \triangleq s_0 \cdot (s_2^+ \cdot s_6^* + s_3 \cdot s_6^+)$*. The set* X *contains* $s_0 + s_0 \cdot s_3 + s_0 \cdot (s_1 \cdot s_5)^* \cdot s_1 + s_0 \cdot (s_1 \cdot s_5)^+ + s_0 \cdot (s_1 \cdot s_5)^+ \cdot s_8^+$*. Finally, the set of suicide strategies is* $s_0 + s_0 \cdot s_3$.

Now, we evaluate the results of function f *for each history in* X*. First, we set* $\mathsf{f}_0(\rho)(j) = 0$, $\forall \rho \in \mathrm{Hst}$ *and* $\forall j \geq 0$*. For all* $k > 0$*,* $i \geq 0$*, and the history* $s_0 s_3$*, we have that*

$$\mathsf{f}_k(s_0 s_3)(i) \triangleq \begin{cases} 0, & \text{if } i > 0; \\ 1, & \text{otherwise}. \end{cases}$$

For all $k > 0$*,* $i \geq 0$*, and* $\rho \in s_0 \cdot (s_1 \cdot s_5)^+ \cdot s_8^+$*, we have that*

$$\mathsf{f}_k(\rho)(i) \triangleq \begin{cases} 0, & \textit{if } i > 0; \\ k, & \textit{otherwise}. \end{cases}$$

For all $k > 0$, $i \geq 0$, and $\rho \in s_0 \cdot (s_1 \cdot s_5)^+$, we have that

$$\mathsf{f}_k(\rho)(i) \triangleq \begin{cases} 0, & \textit{if } k < (2i) + 1; \\ k - ((2i) + 1), & \textit{otherwise}. \end{cases}$$

For all $k > 0$, $i \geq 0$, and $\rho \in s_0 \cdot (s_1 \cdot s_5)^* \cdot s_1$, we have that

$$\mathsf{f}_k(\rho)(i) \triangleq \begin{cases} 0, & \textit{if } k < (2i) \textit{ or } i = 0; \\ k - (2i), & \textit{otherwise}. \end{cases}$$

For all $k > 0$, $i \geq 0$, and the history s_0, we have that

$$\mathsf{f}_k(s_0)(i) \triangleq \begin{cases} 0, & \textit{if } k < (2i) + 1 \textit{ and } i > 0 \textit{ or } k < 2 \textit{ and } i = 0; \\ 1, & \textit{if } k \geq 2 \textit{ and } i = 0; \\ k - ((2i) + 1), & \textit{otherwise}. \end{cases}$$

Now, we illustrate the results of fixpoint $\mathsf{f}^\star$. For all $i \geq 0$ and the history $s_0 s_3$, we have that

$$\mathsf{f}^\star(s_0 s_3)(i) \triangleq \begin{cases} 0, & \textit{if } i > 0; \\ 1, & \textit{otherwise}. \end{cases}$$

For all $i \geq 0$ and $\rho \in s_0 \cdot (s_1 \cdot s_5)^+ \cdot s_8^+$, we have that

$$\mathsf{f}^\star(\rho)(i) \triangleq \begin{cases} 0, & \textit{if } i > 0; \\ \omega, & \textit{otherwise}. \end{cases}$$

For all $i \geq 0$ and $\rho \in s_0 \cdot (s_1 \cdot s_5)^+$, we have that

$$\mathsf{f}^\star(\rho)(i) \triangleq \omega$$

For all $i \geq 0$ and $\rho \in s_0 \cdot (s_1 \cdot s_5)^* \cdot s_1$, we have that

$$\mathsf{f}^\star(\rho)(i) \triangleq \begin{cases} 0, & \textit{if } i = 0; \\ \omega, & \textit{otherwise}. \end{cases}$$

For all $k > 0$, $i \geq 0$, and history the s_0, we have that

$$\mathsf{f}^\star(s_0)(i) \triangleq \begin{cases} 1, & \textit{if } i = 0; \\ \omega, & \textit{otherwise}. \end{cases}$$

Finally, we evaluate the results of grading function. For all $j \geq 0$ and the history $s_0 s_3$, we have that

$$\mathsf{G}^a_\psi(s_0 s_3)(j) \triangleq \begin{cases} 1, & \textit{if } j = 0; \\ 2, & \textit{otherwise}. \end{cases}$$

For all $j \geq 0$ *and* $\rho \in s_0 \cdot (s_1 \cdot s_5)^+ \cdot s_8^+$*, we have that*

$$\mathsf{G}^a_\psi(\rho)(j) \triangleq \omega$$

For all $\rho \in s_0 \cdot (s_1 \cdot s_5)^+ \cup s_0 \cdot (s_1 \cdot s_5)^* \cdot s_1 \cup \{s_0\}$*, we have the same result of function* $\mathsf{f}^\star$*,* i.e.*,* $\mathsf{G}^a_\psi(\rho)(j) \triangleq \mathsf{f}^\star(\rho)(j)$ *for all* $j \geq 0$*.*

3.6 Model Checking

We finally describe a solution of the model-checking problem for the fragment of GSL[1G, 2AG], in which all temporal properties are used as in ATL. This means that we only admit *simple temporal properties*, *i.e.*, $\varphi_1 \mathtt{U} \varphi_2$, $\varphi_1 \mathsf{R} \varphi_2$, and $\mathtt{X}\varphi$, where φ_1, φ_2, and φ are sentences. This fragment, called Vanilla GSL[1G, 2AG], is in relation with GSL[1G, 2AG], as CTL and ATL are with CTL* and ATL*, respectively.

The idea here is to exploit the characterization of the grading function stated in Lemma 3.5.2 in order to verify whether a game structure $\mathcal{G}$ satisfies a sentence $\varphi = \langle\!\langle \mathtt{x} \geq i \rangle\!\rangle [\![\overline{\mathtt{x}} \leq j]\!] (\alpha, \mathtt{x})(\overline{\alpha}, \overline{\mathtt{x}}) \psi$, while avoiding the naive calculation of the least fixpoint F^α_ψ, which requires an infinite calculation due to the cycles of the structure.

Fortunately, due to the simplicity of the temporal property ψ, we have that the four sets W_ψ, $\mathrm{W}_{\neg\psi}$, X, and S previously introduced are memoryless, *i.e.*, if a history belongs to them, every other history ending in the same state is also a member of these sets. Therefore, we can focus only on states by defining $\mathrm{W}_\psi \triangleq \{s \in \mathrm{St} : \mathcal{G}, s \models \mathtt{A}\psi\}$, $\mathrm{W}_{\neg\psi} \triangleq \{s \in \mathrm{St} : \mathcal{G}, s \models \mathtt{A}\neg\psi\}$, $\mathrm{X} \triangleq \mathrm{St} \setminus (\mathrm{W}_\psi \cup \mathrm{W}_{\neg\psi})$, and $\mathrm{S} \triangleq \{s \in \mathrm{St} : \mathcal{G}, s \models \mathtt{E}(\alpha \mathtt{U} \mathtt{A} \neg\psi)\}$ via very simple CTL properties. Observe that we use α and $\overline{\alpha}$ as labeling of a state to recognize its owner. Intuitively, W_ψ and $\mathrm{W}_{\neg\psi}$ contain the states from which agents α and $\overline{\alpha}$ can ensure, independently from the adversary, the properties ψ and $\neg\psi$, respectively. The set X, instead, contains the states on which we have still to determine the number of strategies at disposal of the two agents. Finally, S maintains the suicide states, *i.e.*, those states from which α can commit suicide by autonomously reaching $\mathrm{W}_{\neg\psi}$. In addition, since at most j strategies of $\overline{\alpha}$ can be avoided while reasoning on the sentence φ, we need just to deal with functions in the set $\Gamma \triangleq [0, j] \to (\mathbb{N} \cup \{\omega\})$ instead of $\Gamma \triangleq \mathbb{N} \to (\mathbb{N} \cup \{\omega\})$. Consequently, the functor $\mathsf{F}^\alpha_\psi : (\mathrm{X} \to \Gamma) \to (\mathrm{X} \to \Gamma)$ can be redefined as follows:

$$\mathsf{F}^\alpha_\psi(\mathsf{f})(s)(h) \triangleq \begin{cases} \sum_{s' \in \mathsf{suc}(s) \cap \mathrm{X}} \mathsf{f}(s')(0) + |\mathsf{suc}(s) \cap \mathrm{W}_\psi|, & \text{if } s \in \mathrm{St}_\alpha \text{ and } h = 0; \\ \sum_{s' \in \mathsf{suc}(s) \cap \mathrm{X}} \mathsf{f}(s')(h), & \text{if } s \in \mathrm{St}_\alpha \text{ and } h > 0; \\ \sum_{\mathsf{c} \in \mathrm{C}(s)(h)} \prod_{s' \in \mathsf{dom}(\mathsf{c})} \mathsf{f}(s')(\mathsf{c}(s')), & \text{otherwise;} \end{cases}$$

where $\mathsf{suc}(s) = \{s' \in \mathrm{St} : (s, s') \in Ed\}$ and $\mathsf{C}(s)(i) \subseteq (\mathsf{suc}(s) \cap \mathrm{X}) \rightharpoonup \mathbb{N}$ contains all partial functions $\mathsf{c} \in \mathsf{C}(s)(i)$ for which α owns a suicide strategy on the states not in their domains, *i.e.*, $(\mathsf{suc}(s) \cap \mathrm{X}) \setminus \mathsf{dom}(\mathsf{c}) \subseteq \mathrm{S}$, and the sum of all values assumed by c plus the number of successors that are neither surely winning nor contained in the domain of c equals to i, *i.e.*, $i = \sum_{s' \in \mathsf{dom}(\mathsf{c})} \mathsf{c}(s') + |\mathsf{suc}(s) \setminus (\mathrm{W}_\psi \cup \mathsf{dom}(\mathsf{c}))|$. Similarly, the grading function $\mathsf{G}^\alpha_\psi : \mathrm{X} \to \Gamma$ becomes

$$\mathsf{G}^\alpha_\psi(s)(h) \triangleq \sum_{l=0}^{h} \mathsf{f}^\star(s)(l) + \begin{cases} 1, & \text{if } s \in \mathrm{S} \text{ and } h \geq 1; \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathsf{f}^\star$ is the least fixpoint of F^α_ψ. Observe that the existence of such a fixpoint can be proved in the same way of Lemma 3.5.1, where the set of functions D is $\mathrm{D} \triangleq \mathrm{X} \to \Gamma$, where $\Gamma \triangleq [0, j] \to (\mathbb{N} \cup \{\omega\})$. Unfortunately, these redefinitions are not enough by their own to ensure that the fixpoint calculation can be done in a finite, possibly small, number of iterations of the functor. This is due to two concomitant factors: the functions in Γ have an infinite codomain and the game structure $\mathcal{G}$ might have cycles inside. In order to solve such a problem, we make use of the following observation. Suppose that agent α has at least one strategy on one of its states $s \in \mathrm{St}_\alpha$ against all strategies of its opponent $\overline{\alpha}$ that is also part of a cycle in which no state of $\overline{\alpha}$ is adjacent to a state belonging to the set $\mathrm{W}_{\neg\psi}$. Then, α can use this cycle from s to construct an infinite number of nonequivalent strategies, by simply pumping-up the number of times he decides to traverse it before following the previously found strategy. Therefore, in this case, we avoid to compute the infinite number of iterations required to reach the fixpoint, by directly assuming ω as value. Formally, we introduce the functor $\mathsf{I} : (\mathrm{X} \to \Gamma) \to (\mathrm{X} \to \Gamma)$ defined as follows, where $\mathrm{L} \subseteq \mathrm{St}_\alpha$ denotes the set of α-states belonging to a cycle of the above kind: $\mathsf{I}(\mathsf{f})(s)(h) = \omega$, if $s \in \mathrm{L}$ and $\mathsf{f}(s)(h) > 0$, and $\mathsf{I}(\mathsf{f})(s)(h) = \mathsf{f}(s)(h)$, otherwise, for all $s \in \mathrm{St}$ and $h \in [0, j]$. It can be proved that $\mathsf{f}^\star = (\mathsf{I} \circ \mathsf{F}^\alpha_\psi)(\mathsf{f}^\star)$ iff $\mathsf{f}^\star = \mathsf{F}^\alpha_\psi(\mathsf{f}^\star)$, *i.e.*, the functor obtained by composing I and F^α_ψ has exactly the same least fixpoint of F^α_ψ. Moreover, $\mathsf{f}^\star = (\mathsf{I} \circ \mathsf{F}^\alpha_\psi)^n(\mathsf{f}_\mathrm{o})$ where $j \cdot |\mathcal{G}| \leq n$ and f_o is the zero function, *i.e.*, $\mathsf{f}_\mathrm{o}(s)(h) = 0$, for all $s \in \mathrm{X}$ and $h \in [0, j]$. Hence, we can compute $\mathsf{f}^\star$ in a number of iterations of $\mathsf{I} \circ \mathsf{F}^\alpha_\psi$ that is linear in both the degree j and the size of $\mathcal{G}$. Finally, it is not hard to see that the computation of the sets L can be done in polynomial time.

As an example of an application of the model-checking procedure, consider the two-agent turn-based game structure $\mathcal{G}$ depicted in Figure 3.7, with the circle states ruled by α, the square ones by its opponent $\overline{\alpha}$, and where $\mathtt{s}_5$ and $\mathtt{s}_8$ are labeled by the atomic proposition $\mathtt{p}$. Also, consider the vanilla GSL[1G, 2AG] sentence $\varphi = \langle\!\langle \mathtt{x} \geq i \rangle\!\rangle [\![\overline{\mathtt{x}} \leq j]\!] (\alpha, \mathtt{x})(\overline{\alpha}, \overline{\mathtt{x}})\mathtt{Fp}$. First, we need to compute the preliminary sets of states $\mathrm{W}_{\mathtt{Fp}} = \{\mathtt{s}_5, \mathtt{s}_8\}$ (the light-gray area), $\mathrm{W}_{\neg\mathtt{Fp}} = \{\mathtt{s}_3, \mathtt{s}_6\}$ (the dark-gray area), $\mathrm{X} = \{\mathtt{s}_0, \mathtt{s}_1, \mathtt{s}_2, \mathtt{s}_4, \mathtt{s}_7\}$ (the white area partitioned into strong-connected components), $\mathrm{S} = \{\mathtt{s}_0, \mathtt{s}_2\}$, and $\mathrm{L} = \{\mathtt{s}_7\}$. Now, we can evaluate the fixpoint $\mathsf{f}^\star$ of the functor $\mathsf{I} \circ \mathsf{F}^\alpha_\psi$ that can be obtained, due to the topology of $\mathcal{G}$, after $2(j + 1)$

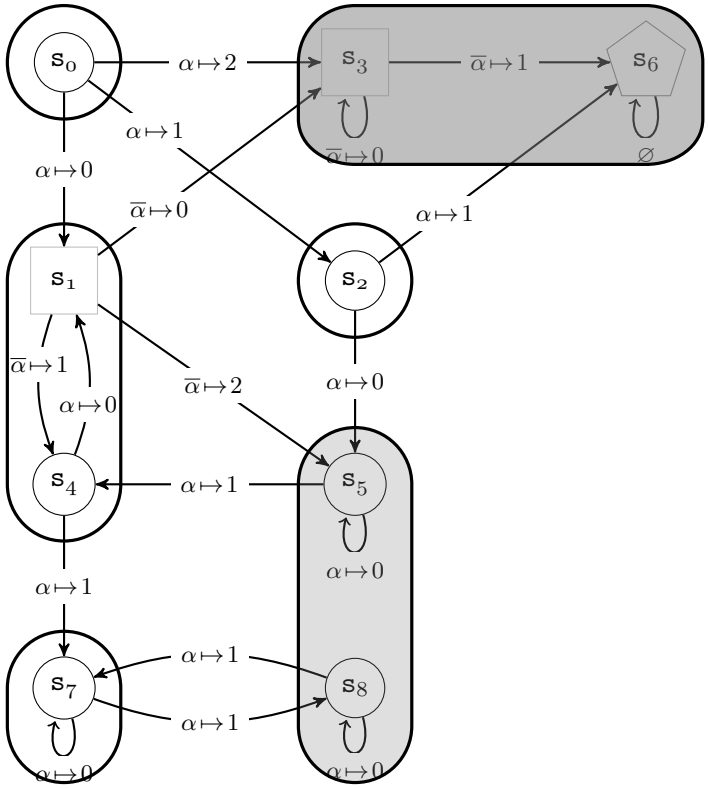


Figure 3.7: A two-player turn-based game structure.

iterations, *i.e.*, $\mathsf{f}^\star = (\mathsf{I} \circ \mathsf{F}^\alpha_\psi)^{2(j+1)}(\mathsf{f}_\mathrm{o})$. Indeed, at the first one, the values on the states $\mathtt{s}_2$ and $\mathtt{s}_7$ are stabilized to $\mathsf{f}^\star(\mathtt{s}_2)(0) = 1$, $\mathsf{f}^\star(\mathtt{s}_7)(0) = \omega$, and $\mathsf{f}^\star(\mathtt{s}_2)(h) = \mathsf{f}^\star(\mathtt{s}_7)(h) = 0$, for all $h \in [1, j]$. After $2j$ iterations, we obtain $\mathsf{f}^\star(\mathtt{s}_1)(0) = 0$, $\mathsf{f}^\star(\mathtt{s}_1)(h) = \omega$, for all $h \in [1, j]$, and $\mathsf{f}^\star(\mathtt{s}_4)(h) = \omega$, for all $h \in [0, j]$. By computing the last iteration, we derive $\mathsf{f}^\star(\mathtt{s}_\mathrm{o})(0) = 1$ and $\mathsf{f}^\star(\mathtt{s}_\mathrm{o})(h) = \omega$, for all $h \in [1, j]$. Note that $2(j + 1)$ is exactly the sum $1 + 2j + 1$ of iterations that the components of the longest chain $\{\mathtt{s}_7\} < \{\mathtt{s}_1, \mathtt{s}_4\} < \{\mathtt{s}_\mathrm{o}\}$ need in order to stabilize the values on their states. Finally, we can verify whether $\mathcal{G} \models \varphi$, by computing the grading function $\mathsf{G}^\alpha_{\mathtt{Fp}}$ at s_o, whose values are $\mathsf{G}^\alpha_{\mathtt{Fp}}(s_\mathrm{o})(0) = 1$ and $\mathsf{G}^\alpha_{\mathtt{Fp}}(s_\mathrm{o})(h) = \omega$, for all $h \in [1, j]$. Thus, $\mathcal{G} \models \varphi$ iff $i = 1$ or $j > 0$.

In order to obtain a PTIME procedure, we have also to ensure that each evaluation of the composed functor $\mathsf{I} \circ \mathsf{F}^\alpha_\psi$ can be computed in PTIME *w.r.t.* the above mentioned parameters. Actually, the whole I and the first two items of F^α_ψ can easily be calculated in linear time. The third item, instead, may require a sum of an exponential number of elements. Indeed, due to all possible ways a degree j can split among the successors of a state s, the corresponding set $\mathrm{C}(s)(j)$ may contain an exponential number of functions. To avoid this, by exploiting a technique similar to the one proposed in [BMM10, BMM12], we linearly transform a game structure into an equivalent one where all states ruled by $\overline{\alpha}$ have degree at most 2. Formally, starting from the CGS $\mathcal{G} \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \mathsf{tr}, \mathsf{ap}, s_\mathrm{o} \rangle$, we construct the equivalent $\mathcal{G}' \triangleq$

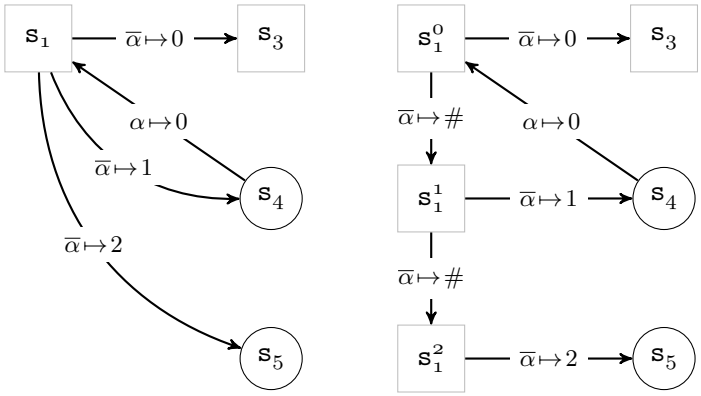


Figure 3.8: Degree transformation.

$\langle \mathrm{AP}', \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}', \mathsf{tr}', \mathsf{ap}', s_I \rangle$. The set of states is defined as follows: $\mathrm{St}' \triangleq \mathrm{St}_\alpha \cup \mathrm{St}_{\overline{\alpha}}$ with $\mathrm{St}_\alpha = \{s^0 | s \in \mathrm{St} \wedge \mathsf{own}(s) = \alpha\}$ and $\mathrm{St}_{\overline{\alpha}} = \{s^i | s \in \mathrm{St} \wedge 0 \leq i < |\mathsf{suc}(s)| \wedge \mathsf{own}(s) = \overline{\alpha}\}$, where $s^0, s^1, \ldots, s^{|\mathsf{suc}(s)|-1}$ are fresh states representing $|\mathsf{suc}(s)|$ different copies of state s. The labeling function is so defined: $\mathsf{ap}'(s^0) = \mathsf{ap}(s)$ and $\mathsf{ap}'(s^i) = \{\#\}$, for each $0 \leq i < |\mathsf{suc}(s)|$. Let $\mathsf{en} : \mathrm{St} \times \mathrm{Dc} \rightharpoonup [0, |\mathsf{suc}(s) - 1|[$ be a partial function that returns the index of the decision such that it is injective and $\mathsf{dom(en)} = \{(s, \delta) | \delta \in \mathsf{dc}(s)\}$, we define $\mathsf{dc}'(s^i) \triangleq \{\overline{\alpha} \to \#\} \cup \{\delta\}$ if $i \in [0, |\mathsf{suc}(s)|[$ and $\mathsf{en}(s, \delta) = i$, while $\mathsf{dc}'(s^i) \triangleq \delta$ if $i = |\mathsf{suc}(s) - 1|$ and $\mathsf{en}(s, \delta) = i$. At this point, we can define the transition function as follows:

$$\mathsf{tr}'(\delta)(s^i) \triangleq \begin{cases} s^{i+1}, & \text{if } s^i \in \mathrm{St}_{\overline{\alpha}} \text{ and } \delta_{\overline{\alpha}} = \#; \\ t^0, & \text{otherwise, where } t = \mathsf{tr}(\delta)(s)\ . \end{cases}$$

In this way, the cardinality of $\mathrm{C}(s)(j)$ is bounded by j. For example, consider the left part of Figure 3.8 representing the substructure of the previous game structure $\mathcal{G}$ induced by the state s_1 together with its three successors. It is not hard to see that we can replace it, in $\mathcal{G}$, by the binary graph at its right, without changing the number of strategies that the two agents have at their disposal.

Theorem 3.6.1 (Model Checking) *The model-checking problem for* Vanilla GSL[1G, 2AG] *is* PTIME *in both the size of the game structure and the sentence. Moreover it is* PTIME-HARD w.r.t. *both the data and combined complexity.*

Proof. Suppose we want to verify that the vanilla GSL[1G, 2AG] sentence $\varphi = \langle\!\langle \mathtt{x} \geq i \rangle\!\rangle [\![\overline{\mathtt{x}} \leq j]\!] (\alpha, \mathtt{x})(\overline{\alpha}, \overline{\mathtt{x}})\psi$ holds on a binary CGS $\mathcal{G}$ (the general case easily follows by induction on the syntactic structure of the sentence, as in the classic ATL model-checking approach, and by linearly transforming any CGS into a binary one). First observe that the computation of the sets of surely-winning states W_ψ, surely-loosing states $\mathrm{W}_{\neg\psi}$, undetermined states

$\mathrm{X} = \mathrm{St} \setminus (\mathrm{W}_{\psi} \cup \mathrm{W}_{\neg\psi})$, and suicide states S can be computed in linear time, since they are defined by means of simple CTL properties. Moreover, let $\mathrm{Z} \subseteq \mathrm{St}_{\overline{\alpha}}$ be the set of $\overline{\alpha}$-states adjacent to some surely-loosing state, and $\mathrm{L} \subseteq \mathrm{St}_{\alpha}$ the set of α-states of all cycles whose nodes do not belong to Z. Note that the elements of Z can be determined in liner time *w.r.t.* the number of moves of the CGS $\mathcal{G}$. Similarly, the sets L can be computed by applying the classic cycle-detection procedure based on DFS on the graph induced by the nodes in $\mathrm{St} \setminus \mathrm{Z}$. At this point, we can compute the least fixpoint $\mathsf{f}^{\star}$ of the functor $\mathsf{I} \circ \mathsf{F}^{\alpha}_{\psi}$ in a number of steps that is bounded by the product of j with the number of moves in $\mathcal{G}$, where every step only requires polynomial time. As we show later, $\mathsf{f}^{\star}$ is the least fixpoint for $\mathsf{F}^{\alpha}_{\psi}$ as well. Therefore, the grading function $\mathsf{G}^{\alpha}_{\psi}$ is immediately derived. Finally, $\mathcal{G} \models \varphi$ iff the conditions stated in Lemma 3.5.2 hold.

For the lower bound, observe that the PTIME hardness *w.r.t.* the size of the game is derived from the fact that classic reachability games [Imm81] are subsumed. Instead, the hardness *w.r.t.* the combined complexity follows as GSL[1G, 2AG] subsumes CTL [Sch02].

It remains to verify that the least fixpoint $\mathsf{f}^{\star}$ of $\mathsf{F}^{\alpha}_{\psi}$ is exactly the least fixpoint of $\mathsf{I} \circ \mathsf{F}^{\alpha}_{\psi}$, which can be computed in polynomial time. Since $\mathsf{F}^{\alpha}_{\psi}(\mathsf{f}^{\star}) = \mathsf{f}^{\star}$, to prove that $(\mathsf{I} \circ \mathsf{F}^{\alpha}_{\psi})(\mathsf{f}^{\star}) = \mathsf{f}^{\star}$, it is enough to show that $\mathsf{I}(\mathsf{f}^{\star}) = \mathsf{f}^{\star}$. Indeed, $(\mathsf{I} \circ \mathsf{F}^{\alpha}_{\psi})(\mathsf{f}^{\star}) = \mathsf{I}(\mathsf{F}^{\alpha}_{\psi}(\mathsf{f}^{\star})) = \mathsf{I}(\mathsf{f}^{\star})$. Now, let $s \in \mathrm{X}$ and $h \in [0, h]$. By definition of the functor I, we have that, if $s \notin \mathrm{L}$, then $\mathsf{I}(\mathsf{f}^{\star})(s)(h) = \mathsf{f}^{\star}(s)(h)$, else either $\mathsf{I}(\mathsf{f}^{\star})(s)(h) = \omega$ and $\mathsf{f}^{\star}(s)(h) > 0$ or $\mathsf{I}(\mathsf{f}^{\star})(s)(h) = \mathsf{f}^{\star}(s)(h) = 0$. Therefore, we have only to show that, if $\mathsf{f}^{\star}(s)(h) > 0$, then $\mathsf{f}^{\star}(s)(h) = \omega$, where $s \in \mathrm{L}$. Since $s \in \mathrm{L}$, there exists a cycle $s = s_0, s_1, \ldots, s_n = s$, where all nodes do not belong to Z_h. In particular, by induction on the length n, we can show that there necessarily exists an index $i \in [0, n[$ such that $s_i \in \mathrm{St}_{\alpha}$ has a successor s' different from s_{i+1}, where $\mathsf{f}^{\star}(s')(h) > 0$. Otherwise, we would have the existence of a fixpoint $\mathsf{f}^{\star\prime}$ for the functor $\mathsf{F}^{\alpha}_{\psi}$, where $\mathsf{f}^{\star\prime}(s_i)(h) = 0$, which contradicts the fact that $\mathsf{f}^{\star}$ is the least fixpoint of $\mathsf{F}^{\alpha}_{\psi}$. Moreover, by direct inspection of the definition of $\mathsf{F}^{\alpha}_{\psi}$, we have that $\mathsf{f}^{\star}(s)(h) = \mathsf{f}^{\star}(s_0)(h) \geq \mathsf{f}^{\star}(s_1)(h) \geq \ldots \geq \mathsf{f}^{\star}(s_i)(h) \geq \mathsf{f}^{\star}(s')(h) + \mathsf{f}^{\star}(s_{i+1})(h) \geq \mathsf{f}^{\star}(s')(h) + \mathsf{f}^{\star}(s_{i+2})(h) \geq \ldots \geq \mathsf{f}^{\star}(s')(h) + \mathsf{f}^{\star}(s_n)(h) = \mathsf{f}^{\star}(s')(h) + \mathsf{f}^{\star}(s)(h)$, *i.e.*, $\mathsf{f}^{\star}(s)(h) \geq \mathsf{f}^{\star}(s')(h) + \mathsf{f}^{\star}(s)(h) \geq 1 + \mathsf{f}^{\star}(s)(h)$. Hence, we necessarily have $\mathsf{f}^{\star}(s)(h) = \omega$.

Finally, to show that $\mathsf{f}^{\star}$ can be computed in polynomial time, we prove that $\mathsf{f}^{\star} = (\mathsf{I} \circ \mathsf{F}^{\alpha}_{\psi})^n(\mathsf{f}_0)$, where $n = j \cdot |\mathcal{G}|$ and f_0 is the zero function, *i.e.*, $\mathsf{f}_0(s)(h) = 0$, for all $s \in \mathrm{X}$ and $h \in [0, j]$. Let $\mathsf{f}^i = (\mathsf{I} \circ \mathsf{F}^{\alpha}_{\psi})^i(\mathsf{f}_0)$, for $i \in]0, n]$. It is easy to observe that $\mathsf{f}^i(s)(0) = 0$, for all states $s \in \mathrm{Z}$ and indexes $i \in [0, n]$. This is due to the fact that the set of functions $\mathrm{C}(s)(0)$ used in the definition of $\mathsf{F}^{\alpha}_{\psi}$ is necessarily empty. Therefore, after $|\mathcal{G}|$ iterations, we have that all values of $\mathsf{f}^{\star}(s)(0)$ are determined. Indeed, all cycles passing through Z cannot pump up the values of the corresponding nodes, while those avoiding Z, thanks to the functor I, immediately reach the value ω as soon as they are positive. The same reasoning applies, in general, for the computation of the values $\mathsf{f}^{\star}(s)(h)$ that are determined after at most $h|\mathcal{G}|$

iterations, since they only depend on the values $\mathsf{f}^{\star}(s')(h')$, where $h' \leq h$. □

3.7 Discussion

In multi-agent systems general questions to be investigated are: *"is there a winning strategy?"* or *"is the game surely winning?"* (*i.e.*, no matter which strategy the agent can play). In the years, several logics suitable for the strategic reasoning have been introduced and, by means of existential and universal modalities, this kind of questions has been addressed [AHK02]. However, these logics are not able to address quantitative aspects such as *"what is the number of winning strategies an agent can play?"* or, in general, to determine the *success rate of a game* [MMS15]. These questions are critical in dealing with solution concepts [Mye91] and in open-system verification [FMP08].

In this chapter, we have introduced and studied GSL, an extension of Strategy Logic with graded modalities. The use of a powerful formalism such as Strategy Logic ensures the ability of dealing with very intricate game scenarios [MMV10a]. The obvious drawback of this is a considerable amount of work on solving any related question [FAGV12]. One of the main difficulties we have faced in GSL has been the definition of the right methodology to count strategies. To this aim, we have introduced a suitable equivalence relation over strategy profiles based on the strategic behavior they induce and studied its robustness. Also, we have provided arguments and some examples along the chapter to give evidence of the usefulness of GSL and the suitability of the proposed counting.

In order to provide results of practical use, we have investigated basic questions over a restricted fragment of GSL. Precisely we have considered the case in which the graded modalities are applied to the *vanilla* restriction of the one-goal fragment of SL [FAGV12]. The resulting logic, named *Vanilla* SL$[1\mathrm{G}]$, has been investigated in the turn-based setting. We have obtained positive results about determinacy and showed that the related model-checking problem is PTIME-COMPLETE.

The framework and the results presented in this work open for several future work questions. First, it would be worth investigating the extension of existing formal verification tools such as MCMAS [LR06b] with our results. We recall that MCMAS, originally developed for the verification for multi-agent models with respect to specification given in ATL [LR06b], has been recently extended to handle Strategy Logic specifications [ČLMM14, ČLM15]. Under our formalism it is possible to check, in a single evaluation process, that more than one strategy gives a fault and possibly correct all these errors. This in a way similar as the verification tool NuSMV has been extended to deal with *graded*-CTLverification [FNP09a].

Another research direction regards investigating the graded extension of other formalism for the strategic reasoning such as ATL *with context* [BLLM09, LLM10], as well as, for the sake of completeness, to determine the complexity of the model checking problem with

respect to other fragments of Strategy Logic [MMS14b, MMS14a].

Finally, it would be really interesting to address the satisfiability for GSL[1G] too, by generalizing the solution procedure developed for SL[1G] [FAGV12]. However, we want to observe that, the technical tools described in this article are not powerful enough to solve this problem, since this also needs a bounded-width tree model property. So, further work is still required. Moreover, the procedure exploited for graded CTL [BMM09, BMM10, BMM12] cannot easily be applied to GSL[1G], due to the fact that the binary-tree unraveling used there would modify the way the strategies are valuated as equivalent.

CHAPTER 4

Graded Modalities in Strategy Logic

Contents

4.1 Introduction

Strategy Logic (SL) is a powerful formalism for reasoning about strategies in multi-agent systems [MMV10a, MMPV14]. Strategies tell an agent what to do — they are functions that prescribe an action based on the history. The key idea in SL is to treat strategies as first-order objects. A strategy x can be quantified existentially $\langle\langle x \rangle\rangle$ (read: there exists a strategy x) and universally $[\![x]\!]$ (read: for all strategies x). Furthermore, strategies are not intrinsically glued to specific agents: the *binding* operator (α, x) allows one to bind an agent α to the strategy x. SL strictly subsumes several other logics for strategic reasoning including the well known ATL and ATL* [AHK02]. Being a very powerful logic, SL can directly express many solution concepts [CHP10, MMPV14, GHW17, KPV14, Bel15, BLMR17, GMP$^+$17, BMMRV17] among which that a strategy profile $\overline{x}$ is a Nash equilibrium, and thus also the existence of a Nash equilibrium (NE).

The Nash equilibrium is one of the most important concepts in game theory, forming the basis of much of the recent fundamental work in multi-agent decision making. A challenging and important aspect is to establish whether a game admits a *unique* NE [AKH02, PC79, CHS99]. This problem is relevant to the predictive power of NE since, in case there are multiple equilibria, the outcome of the game cannot be uniquely pinned down [SLCB13, ZG11, Pav12]. Unfortunately, uniqueness has mainly been established either for special cost functions [AKH02], or for very restrictive game topologies [ORS93]. Moreover, there is no general theory of when games have unique equilibria that can be applied to different application areas [AKH02].

In this chapter, we address and solve the problem of expressing the uniqueness of certain solution concepts (and NE in particular) in a principled and elegant way, by introducing an extension of SL called GRADEDSL. More specifically, we extend SL by replacing the quantification $\langle\langle x \rangle\rangle$ and $[\![x]\!]$ over strategy variables with *graded quantification over tuples of strategy variables*: $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}$ (read $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}$ as "there exist at least g different tuples $(x_1, \ldots, x_n)$ of strategies") and its dual $[\![x_1, \ldots, x_n]\!]^{<g}$, where $g \in \mathbb{N} \cup \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$. Here, two tuples are different if they are different in some component, and two strategies are different if they disagree on some history. The key is being able to express uniqueness of NE is the combination of quantifying over tuples (instead of singleton variables), and adding counting (in the form of graded modalities).

As far as the expressive power of GRADEDSL concerns, we prove that counting strategies in SL is not possible in general (see Theorem 4.2.1). On the other hand, every formula of SL has an equivalent formula of GRADEDSL formed by replacing every quantifier $\langle\langle x \rangle\rangle$ with $\langle\langle x \rangle\rangle^{\geq 1}$. Additionally, the possibility of quantifying over tuples of strategy variables (rather than single strategies) makes the logic quite expressive.

We address the model-checking problem for GRADEDSL and prove that it is decidable.

We also address the complexity of several fragments of GRADEDSL. First we consider the case in which the g's are restricted to finite cardinals, written GRADED$_{\mathbb{N}}$SL. Then we investigate the graded extension of classic fragments of SL, such as Nested-Goal SL and one-goal SL [MMPV14], while maintaining the restriction of grades over finite cardinals. Roughly speaking, the Nested-Goal restriction encompasses formulas in a special prenex normal form with a particular nested temporal structure that restricts the application of both strategy quantifiers and agent bindings; further, the one-goal restriction is obtained by forbidding any nesting and Boolean operation over bindings (see Section 4.2.4 for details).

We show that the complexity of the model-checking problem for GRADED$_{\mathbb{N}}$SL is no harder than for SL, i.e., it is non-elementary in the nesting depth of quantifiers. In particular, we show that model checking GRADED$_{\mathbb{N}}$SL formulas with a nesting depth $k > 0$ of blocks of quantifiers (a block of quantifiers is a maximally-consecutive sequence of quantifiers of the same type, i.e., either all existential, or all universal) is in $(k+1)$EXPTIME, and that for the special case where the formula starts with a block of quantifiers, it is in kEXPTIME. Since many natural formulas contain a very small number of quantifiers, the complexity of the model-checking problem is not as bad as it seems. Specifically, several solution concepts can be expressed as SL formulas with a small number of quantifiers[CHP10, MMPV14, GHW17, KPV14, Bel15]. Since the existence of a NE, and the fact that there is at most one NE, can be expressed in GRADED$_{\mathbb{N}}$SL using simple formulas (assuming that the agents' goals are given as LTL formulas) we are able to conclude that the problem of checking the uniqueness of a NE can be solved in 2EXPTIME. Previously, it was known that existence of NE can be checked in 2EXPTIME [MMPV14]. Thus, GRADEDSL is the first logic that can solve the existence and uniqueness of NE (as well as many other solution concepts) in 2EXPTIME.

Concerning the graded Nested-Goal fragment, namely GRADEDSL[NG], we show that, in case the g's are restricted to finite cardinals, it has the same model-checking complexity as Nested-Goal SL, i.e., non-elementary in the *alternation number* of the quantifiers appearing in the formula. For the one-goal fragment, namely GRADED$_{\mathbb{N}}$SL[1G], the model checking problem is instead 2EXPTIME-COMPLETE. All model checking complexities reported so far refer to the size of the formula.

Related work. The importance of solution concepts, verifying a unique equilibrium, and the relationship with logics for strategic reasoning is discussed above. We now give some highlights from the long and active investigation of graded modalities in the formal verification community.

Graded modalities were first studied in modal logic [Fin72] and then exported to the field of knowledge representation to allow quantitative bounds on the set of individuals satisfying a given property. Specifically, they were considered as *counting quantifiers* in first-order logics [GOR97] and *number restrictions* in *description logics* [HB91]. *Graded μ-calculus*, in which immediate-successor accessible worlds are counted, was introduced to reason about

graded modal logic with fixed-point operators [KSV02]. Recently, the notion of graded modalities was extended to count the number of paths in the branching-time temporal logic formulas CTL and CTL* [BMM12, RAM15].

Graded strategy quantifiers were studied in [MMMS15, FNP09a]. However, in contrast to our work, both of these works have an intricate way of counting. The work in [MMMS15] introduced a graded extension of SL, called GSL. This logic gives a semantic definition for when two strategies should be considered equal, and counts the number of equivalence classes. While this intricate approach is justifiable, it leads to a complicated model-checking problem. Indeed, only a very weak fragment of GSL has been solved in [MMMS15], namely the vanilla restriction of the graded version of the one-goal fragment of SL [FAGV12]. There is a common belief that the one-goal fragment is not powerful enough to express the existence of a Nash Equilibrium in concurrent games. The smallest fragment that is known to be able to represent this is the so called Boolean-goal Strategy Logic, whose graded extension (in the GSL sense) has no known solution.[1]

We note that the work in [FNP09a] introduced a graded extension of ATL with two semantics for graded strategy quantifiers. Both of these semantics also employ an intricate equivalence relation on paths and sets of paths.

Finally, we mention that in the verification of reactive systems there is an orthogonal approach called module checking. [2] Module-checking for graded μ-calculus was studied in [ALMS08, FMP08].

4.2 Graded Strategy Logic

In this section we introduce Graded Strategy Logic, which we call GRADEDSL for short.

In the following we use a finite set Vr of *variables*, a finite set Ag of *agents*, and a finite set AP of *atomic propositions (atoms)*. We denote variables by x_i, x_j, etc., agents by α_i, α_j, etc., and atomic propositions by p, q, etc. The assumption that these sets are finite is simply a technical convenience: the model-checking problem (Definition 4.3.1) takes as input formulas and arenas with any number of variables, agents, and atoms.

[1] In [GHW15] it has been shown that, in the restricted case of turn-based structures it is possible to express the existence of Nash equilibria in m^-ATL* [MMV10b], a memory-full variant of ATL* (hence included in one-goal SL), but exponentially more succinct — and thus with a much more expensive model-checking algorithm. As the authors in [GHW15] also state, it is not clear how to extend this result to the concurrent setting, even in the case of two agents.

[2] Module checking is a decision problem proposed in late 1990s to formalize verification of open systems [KVW01]. Recently it has been showed that module checking offers a distinctly different perspective from the problem of model checking [JM14].

4.2.1 Syntax

GRADEDSL extends SL by replacing the singleton strategy quantifiers $\langle\langle x \rangle\rangle$ and $[\![x]\!]$ with the graded (tupled) quantifiers $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}$ and $[\![x_1, \ldots, x_n]\!]^{<g}$, respectively, where $g \in \mathbb{N} \cup \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$ is called the *grade* of the quantifier. Intuitively, these are read as "there exist at least g tuples of strategies $(x_1, \ldots, x_n)$" and "all but less than g many tuples of strategies", respectively. The syntax (α, x) denotes a *binding* of the agent α to the strategy x.

Definition 4.2.1 GRADEDSL formulas *are built inductively by means of the following grammar, where* $p \in \mathrm{AP}$*,* $\alpha \in \mathrm{Ag}$*,* $x, x_1, \ldots, x_n \in \mathrm{Vr}$ *such that* $x_i \neq x_j$ *for* $i \neq j$ *and* $n \in \mathbb{N}$*, and* $g \in \mathbb{N} \cup \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$*:*

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}\varphi \mid (\alpha, x)\varphi.$$

Note that GRADEDSL formulas are defined *w.r.t.* fixed finite sets of atomic propositions AP, agents Ag, and variables Vr.

Notation. Whenever we write $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}$ we mean that $x_i \neq x_j$ for $i \neq j$, i.e., the variables in a tuple are distinct (note that this does not mean that the strategies the variables represent are distinct).

Shorthands are derived as usual. Specifically, $\mathsf{true} \triangleq p \vee \neg p$, $\mathsf{false} \triangleq \neg\mathsf{true}$, $\mathsf{F}\,\varphi \triangleq \mathsf{true}\,\mathsf{U}\,\varphi$, and $\mathsf{G}\,\varphi \triangleq \neg\,\mathsf{F}\,\neg\varphi$. Also, we have that $[\![x_1, \ldots, x_n]\!]^{<g}\varphi \triangleq \neg\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}\neg\varphi$. The operators $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}$ (resp. $[\![x_1, \ldots, x_n]\!]^{<g}$) are called *existential (resp. universal) strategy quantifiers*.

In order to define the semantics, we first define the concept of *free placeholders* in a formula, which refer to agents and variables. Intuitively, an agent or variable is free in φ if it does not have a strategy associated with it (either by quantification or binding) but one is required in order to ascertain if φ is true or not. The definition mimics that for SL [MMPV14]. It is important for defining the model-checking procedure, in particular for the encoding of strategies as trees (Definition 4.3.3).

Definition 4.2.2 *The set of* free agents *and* free variables $free(\varphi) \in 2^{\mathrm{Ag} \cup \mathrm{Vr}}$ *of a* GRADEDSL *formula* φ *is inductively defined as follows:*

- $\mathsf{free}(p) \triangleq \emptyset$*, where* $p \in \mathrm{AP}$*;*
- $\mathsf{free}(\neg\varphi) \triangleq \mathsf{free}(\varphi)$*;*
- $\mathsf{free}(\varphi_1 \vee \varphi_2) \triangleq \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$*;*
- $\mathsf{free}(\mathsf{X}\,\varphi) \triangleq \mathrm{Ag} \cup \mathsf{free}(\varphi)$*;*
- $\mathsf{free}(\varphi_1\,\mathsf{U}\,\varphi_2) \triangleq \mathrm{Ag} \cup \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$*;*

- $\mathsf{free}(\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq g} \varphi) \triangleq \mathsf{free}(\varphi) \setminus \{x_1, \ldots, x_n\}$;
- $\mathsf{free}((\alpha, x)\varphi) \triangleq \begin{cases} \mathsf{free}(\varphi), & \text{if } \alpha \notin \mathsf{free}(\varphi); \\ (\mathsf{free}(\varphi) \setminus \{\alpha\}) \cup \{x\}, & \text{otherwise.} \end{cases}$

A formula φ without free agents (resp., variables), i.e., with $\mathsf{free}(\varphi) \cap \mathrm{Ag} = \emptyset$ (resp., $\mathsf{free}(\varphi) \cap \mathrm{Vr} = \emptyset$), is called agent-closed *(resp.,* variable-closed*). If φ is both agent- and variable-closed, it is called a* sentence*.*

Roughly, the *quantifier rank* of φ is the maximum, over all paths in the parse-tree of φ, of the number of strategy quantifiers that appear on the path, e.g., $\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq g}(\alpha_1, x_1) \ldots (\alpha_n, x_n) \bigwedge_{i=1}^{n} (\langle\!\langle \mathsf{y} \rangle\!\rangle(\alpha_i, \mathsf{y})\psi_i) \rightarrow \psi_i$ has quantifier rank 2 if each ψ_i is quantifier free.

Definition 4.2.3 *The* quantifier rank $\mathsf{qr}(\varphi) \in \mathbb{N}$ *of a* GRADEDSL *formula φ is inductively defined as follows:*

- $\mathsf{qr}(p) \triangleq 0$*, where* $p \in \mathrm{AP}$*;*
- $\mathsf{qr}(\mathsf{OP}\varphi) \triangleq \mathsf{qr}(\varphi)$*, where* $\mathsf{OP} \in \{\neg, \mathsf{X}, \flat\}$*;*
- $\mathsf{qr}(\varphi_1 \mathsf{OP} \varphi_2) \triangleq \mathsf{max}(\mathsf{qr}(\varphi_1), \mathsf{qr}(\varphi_2))$ *where* $\mathsf{OP} \in \{\vee, \mathsf{U}\}$*;*
- $\mathsf{qr}(\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq g} \varphi) \triangleq \mathsf{qr}(\varphi) + 1$.

Roughly, a *quantifier-block of φ* is a maximally-consecutive sequence of quantifiers in φ of the same type (*i.e.*, either all existential, or all universal). The *quantifier-block rank* of a formula is like the quantifier rank except that a quantifier block of j quantifiers contributes 1 instead of j to the count. The formal definition follows:

Definition 4.2.4 *The* quantifier-block rank $\mathsf{qbr}(\varphi) \in \mathbb{N}$ *of a* GRADEDSL *formula φ that uses the shorthand for universal strategy quantifiers is inductively defined as follows:*

- $\mathsf{qbr}(p) \triangleq 0$*, where* $p \in \mathrm{AP}$*;*
- $\mathsf{qbr}(\mathsf{OP}\varphi) \triangleq \mathsf{qbr}(\varphi)$*, where* $\mathsf{OP} \in \{\neg, \mathsf{X}, \flat\}$*;*
- $\mathsf{qbr}(\varphi_1 \mathsf{OP} \varphi_2) \triangleq \mathsf{max}(\mathsf{qbr}(\varphi_1), \mathsf{qbr}(\varphi_2))$ *where* $\mathsf{OP} \in \{\vee, \mathsf{U}\}$*;*
- $\mathsf{qbr}(\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq g} \varphi) \triangleq \mathsf{qbr}(\varphi)$ *if φ begins with an existential strategy quantifier, and* $\mathsf{qbr}(\varphi) + 1$ *otherwise.*
- $\mathsf{qbr}([\![x_1, \ldots, x_n]\!]^{<g} \varphi) \triangleq \mathsf{qbr}(\varphi)$ *if φ begins with a universal strategy quantifier, and* $\mathsf{qbr}(\varphi) + 1$ *otherwise.*

Note that we treat $\neg\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq g} \neg\varphi$ differently to $[\![x_1, \ldots, x_n]\!]^{<g} \varphi$. Thus, one should choose the existential and universal quantifiers judiciously in order to obtain a low quantifier-block rank.

4.2.2 Models

Sentences of GRADEDSL are interpreted over *arenas* [3], just as for ATL and SL [AHK02, MMPV14].

Definition 4.2.5 *An* arena *over fixed sets of atomic proposition* AP *and agents* Ag *is a tuple* $\mathcal{A} \triangleq \langle \mathrm{Ac}, \mathrm{St}, s_I, \mathsf{ap}, \mathsf{tr} \rangle$*, where:*

- Ac *is a finite set of* actions*;*
- St *is a finite set of* states*;*
- $s_I \in \mathrm{St}$ *is the* initial state*;*
- $\mathsf{ap} : \mathrm{St} \to 2^{\mathrm{AP}}$ *is the* labeling function *mapping each state to the set of atomic propositions true in that state;*
- *Let* $\mathrm{Dc} \triangleq \mathrm{Ag} \to \mathrm{Ac}$ *be the set of* decisions*,* i.e.*, functions describing the choice of an action by every agent. Then,* $\mathsf{tr} : \mathrm{Dc} \to (\mathrm{St} \to \mathrm{St})$ *is a* transition function *mapping every decision* $\delta \in \mathrm{Dc}$ *to a function* $\mathsf{tr}(\delta) : \mathrm{St} \to \mathrm{St}$*.*

We will usually take the set Ag of agents to be $\{\alpha_1, \ldots, \alpha_n\}$. A *path (from s)* is a finite or infinite non-empty sequence of states $s_1 s_2 \ldots$ such that $s = s_1$ and for every i there exists a decision δ with $\mathsf{tr}(\delta)(s_i) = s_{i+1}$. Given a path $\pi = s_1 s_2 \ldots$, with $\lambda(\pi)$ we denote the *label of* π as a sequence of sets of atomic propositions $\pi_1, \pi_2, \ldots$ where $\mathsf{ap}(s_1) = \pi_1$, $\mathsf{ap}(s_2) = \pi_2$, and so on. The set of paths starting with s is denoted $\mathrm{Pth}(s)$. The set of finite paths from s, called the *histories (from s)*, is denoted $\mathrm{Hst}(s)$. A *strategy (from s)* is a function $\sigma \in \mathrm{Str}(s) \triangleq \mathrm{Hst}(s) \to \mathrm{Ac}$ that prescribes which action has to be performed given a history. We write $\mathrm{Pth}, \mathrm{Hst}, \mathrm{Str}$ for the set of all paths, histories, and strategies (no matter where they start). We use the standard notion of equality between strategies [LBS08], i.e., $\sigma_1 = \sigma_2$ iff for all $\rho \in \mathrm{Hst}$, $\sigma_1(\rho) = \sigma_2(\rho)$. This extends to equality between two n-tuples of strategies in the natural way, i.e., coordinate-wise. There is a subtlety in this definition, i.e., two strategies are different if they differ on some history ρ, even if that history is not reachable using either of the strategies.

4.2.3 Semantics

As for SL, the interpretation of a GRADEDSL formula requires a valuation of its free placeholders.

Definition 4.2.6 *An* assignment (from s) *is a function* $\chi \in \mathrm{Asg}(s) \triangleq (\mathrm{Vr} \cup \mathrm{Ag}) \to \mathrm{Str}(s)$ *mapping variables and agents to strategies.*

[3] This is sometimes called a Concurrent Game Structure.

We denote by $\chi[e \mapsto \sigma]$, with $e \in \mathrm{Vr} \cup \mathrm{Ag}$ and $\sigma \in \mathrm{Str}(s)$, the assignment that differs from χ only in the fact that e maps to σ. Extend this definition to tuples: for $\overline{e} = (e_1, \ldots, e_n)$ with $e_i \neq e_j$ for $i \neq j$, define $\chi[\overline{e} \mapsto \overline{\sigma}]$ to be the assignment that differs from χ only in the fact that e_i maps to σ_i (for each i).

Since an assignment ensures that all free variables are associated with strategies, it induces a play.

Definition 4.2.7 *Let $\chi \in \mathrm{Asg}(s)$ be an assignment. By* (χ, s)-play *we denote the path $\pi \in \mathrm{Pth}(s)$ such that, for all $i \in \mathbb{N}$, it holds that $\pi_{i+1} = \mathsf{tr}(\mathsf{dc})(\pi_i)$, where $\mathsf{dc}(\alpha) \triangleq \chi(\alpha)(\pi_{\leq i})$, for $\alpha \in \mathrm{Ag}$. The function $\mathsf{play} : \mathrm{Asg} \times \mathrm{St} \to \mathrm{Pth}$, with $\mathsf{dom}(\mathsf{play}) \triangleq \{(\chi, s) : \chi \in \mathrm{Asg}(s)\}$, maps (χ, s) to the (χ, s)-play $\mathsf{play}(\chi, s) \in \mathrm{Pth}(s)$.*

The notation $\pi_{\leq i}$ (resp. $\pi_{<i}$) denotes the prefix of the sequence π of length i (resp. $i-1$). Similarly, the notation π_i denotes the ith symbol of π. Thus, $\mathsf{play}(\chi, s)_i$ is the ith state on the play determined by χ from s.

The following definition of χ_i says how to interpret an assignment χ starting from a point i along the play, i.e., for each placeholder e, take the action the strategy $\chi(e)$ would do if it were given the prefix of the play up to i followed by the current history.

Definition 4.2.8 *For $\chi \in \mathrm{Asg}(s)$ and $i \in \mathbb{N}$, writing $\rho \triangleq \mathsf{play}(\chi, s)_{\leq i}$ (the prefix of the play up to i) and $t \triangleq \mathsf{play}(\chi, s)_i$ (the last state of ρ) define $\chi_i \in \mathrm{Asg}(t)$ to be the assignment from t that maps $e \in \mathrm{Vr} \cup \mathrm{Ag}$ to the strategy that maps $h \in \mathrm{Hst}(t)$ to the action $\chi(e)(\rho_{<i} \cdot h)$.*

The semantics of GRADEDSL mimics the one for SL as given in [MMPV14]. Given an arena $\mathcal{A}$, for all states $s \in \mathrm{St}$ and assignments $\chi \in \mathrm{Asg}(s)$, we now define the relation $\mathcal{A}, \chi, s \models \varphi$, read φ *holds at s in $\mathcal{A}$ under χ*.

Definition 4.2.9 *Fix an arena $\mathcal{A}$. For all states $s \in \mathrm{St}$ and assignments $\chi \in \mathrm{Asg}(s)$, the relation $\mathcal{A}, \chi, s \models \varphi$ is defined inductively on the structure of φ:*

- $\mathcal{A}, \chi, s \models p$ *iff* $p \in \mathsf{ap}(s)$;
- $\mathcal{A}, \chi, s \models \neg\varphi$ *iff* $\mathcal{A}, \chi, s \not\models \varphi$;
- $\mathcal{A}, \chi, s \models \varphi_1 \vee \varphi_2$ *iff* $\mathcal{A}, \chi, s \models \varphi_1$ *or* $\mathcal{A}, \chi, s \models \varphi_2$;
- $\mathcal{A}, \chi, s \models \mathsf{X}\varphi$ *iff* $\mathcal{A}, \chi_1, \mathsf{play}(\chi, s)_1 \models \varphi$;
- $\mathcal{A}, \chi, s \models \varphi_1 \,\mathsf{U}\, \varphi_2$ *iff there is $i \in \mathbb{N}$ such that* $\mathcal{A}, \chi_i, \mathsf{play}(\chi, s)_i \models \varphi_2$ *and, for all $j \in \mathbb{N}$ with $j < i$, it holds that* $\mathcal{A}, \chi_j, \mathsf{play}(\chi, s)_j \models \varphi_1$;
- $\mathcal{A}, \chi, s \models (\alpha, x)\varphi$ *iff* $\mathcal{A}, \chi[\alpha \mapsto \chi(x)], s \models \varphi$;
- $\mathcal{A}, \chi, s \models \langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}\varphi$ *iff there exist g many n-tuples of strategies $\overline{\sigma_i}$ $(0 \leq i < g)$ such that:*

– $\overline{\sigma_i} \neq \overline{\sigma_j}$ *for* $i \neq j$;

– $\mathcal{A}, \chi[\overline{x} \mapsto \overline{\sigma_i}], s \models \varphi$ *for* $0 \leq i < g$ *and* $\overline{x} = (x_1, \ldots, x_n)$.

Intuitively, $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}\varphi$ expresses that the number of distinct tuples of strategies that satisfy φ is at least g.

As usual, if χ and χ' agree on free(φ), then $\mathcal{A}, \chi, s \models \varphi$ if and only if $\mathcal{A}, \chi', s \models \varphi$, i.e., the truth of φ does not depend on the values the assignment takes on placeholders that are not free. Thus, for a sentence φ we write $\mathcal{A}, s \models \varphi$ to mean that $\mathcal{A}, \chi, s \models \varphi$ for some (equivalently, for all) assignments χ. Also, we write $\mathcal{A} \models \varphi$ to mean $\mathcal{A}, s_I \models \varphi$ where s_I is the initial state of $\mathcal{A}$.

4.2.4 Fragments of GRADEDSL

In this section we introduce various syntactic fragments of GRADEDSL. Obviously SL can be considered a fragment of GRADEDSL: note that the GRADEDSL quantifier $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}$ in case $g = 1$ and $n = 1$ has the same semantics as the SL quantifier $\langle\langle x_1 \rangle\rangle$. The next result shows that GRADEDSL is strictly more expressive than SL, i.e., there is a GRADEDSL sentence whose models are not the set of models of any SL sentence.

Theorem 4.2.1 GRADEDSL *is strictly more expressive than* SL.

Proof. Fix $\mathrm{Ag} = \{\alpha\}$, $\mathrm{AP} = \{p\}$, $\mathrm{St} = \{s\}$, $s_I = s$, $\mathsf{ap}(s) = \{p\}$. Define $\mathcal{A} = \langle \mathrm{Ac}_A, \mathrm{St}, s_I, \mathsf{ap}, \mathsf{tr}_A \rangle$, where $\mathrm{Ac}_A = \{0\}$, and $\mathsf{tr}_A(\delta)(s) = s$ for every decision δ; and $\mathcal{B} = \langle \mathrm{Ac}_B, \mathrm{St}, s_I, \mathsf{ap}, \mathsf{tr}_B \rangle$, where $\mathrm{Ac}_B = \{0, 1\}$, and $\mathsf{tr}_B(\delta)(s) = s$ for every decision δ. Thus, each arena consists of a single state with self-loops, the difference being that in $\mathcal{B}$ there are two actions while in $\mathcal{A}$ there is only a single action.

Consider the GRADEDSL formula $\neg\langle\langle x \rangle\rangle^{\geq 2}\mathsf{true}$. Note that $\mathcal{A} \models \neg\langle\langle x \rangle\rangle^{\geq 2}\mathsf{true}$ (since there is only a single strategy in $\mathcal{A}$), while $\mathcal{B} \not\models \neg\langle\langle x \rangle\rangle^{\geq 2}\mathsf{true}$ (since there are at least two, and in fact $2^{\aleph_0}$ many, strategies in $\mathcal{B}$).

Let χ_A be the unique assignment in $\mathcal{A}$, i.e., that maps α and every variable in Vr to the strategy σ defined by $\sigma(h) = 0$ for all histories h. We claim that, for every SL formula φ, if $\mathcal{A}, \chi_A, s \models \varphi$ then, for all assignments χ we have that $\mathcal{B}, \chi, s \models \varphi$. Thus, in particular, no SL sentence can distinguish between $\mathcal{A}$ and $\mathcal{B}$, and the theorem follows.

One can easily prove the claim by induction on the structure of an SL formula. Alternatively, one may note that $\mathcal{A}$ and $\mathcal{B}$ are locally-isomorphic, and thus agree on all SL formulas (see [Mog11, Section 3] for the definition and properties of "local-isomorphism").[4] □

Recall that SL has a few natural syntactic fragments, the most powerful of which is SL[NG] (here "NG" stands for Nested-Goal). Recall that in SL[NG], we require that bindings

[4] We thank an anonymous reviewer for pointing this out.

and quantifications appear in exhaustive blocks. I.e., whenever there is a quantification over a variable in a formula ψ it is part of a consecutive sequence of quantifiers that covers all of the free variables that appear in ψ, and whenever an agent is bound to a strategy then it is part of a consecutive sequence of bindings of all agents to strategies. Also, formulas with free agents are not allowed. We define GRADEDSL[NG] in a similar way, as follows.

A *quantification prefix* over a set $\mathrm{V} \subseteq \mathrm{Vr}$ of variables is a sequence $\wp$ from the set

$$\{\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq g}, [\![x_1, \ldots, x_n]\!]^{<g} : n \in \mathbb{N}, x_1, \ldots, x_n \in \mathrm{V} \wedge g \in \mathbb{N} \cup \{\aleph_0, \aleph_1, 2^{\aleph_0}\}\}^*$$

such that each $x \in \mathrm{V}$ occurs exactly once in $\wp$. A *binding prefix* is a sequence $\flat \in \{(\alpha, x) : \alpha \in \mathrm{Ag} \wedge x \in \mathrm{Vr}\}^*$ such that each $\alpha \in \mathrm{Ag}$ occurs exactly once in $\flat$. We denote the set of binding prefixes by Bn, and the set of quantification prefixes over V by $\mathrm{Qn}(\mathrm{V})$.

Definition 4.2.10 GRADEDSL[NG] *formulas are built inductively using the following grammar, with* $p \in \mathrm{AP}$, $\wp \in \mathrm{Qn}(\mathrm{V})$ *(*$\mathrm{V} \subseteq \mathrm{Vr}$*), and* $\flat \in \mathrm{Bn}$*:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \wp\varphi \mid \flat\varphi,$$

where in the rule $\wp\varphi$ *we require that* φ *is agent-closed and* $\wp \in \mathrm{Qn}(\mathsf{free}(\varphi))$*. In this case we call* $\wp\varphi$ *a* principal *formula.*

Formulas of GRADEDSL[NG] can be classified according to their *alternation number*, i.e., the maximum number of quantifier switches in a quantification prefix. [5] Formally:

Definition 4.2.11 *The* alternation number $\mathsf{alt}(\varphi)$ *of a* GRADEDSL[NG] *formula* φ *is defined as follows:*

- $\mathsf{alt}(p) \triangleq 0$*, where* $p \in \mathrm{AP}$*;*
- $\mathsf{alt}(\mathsf{OP}\varphi) \triangleq \mathsf{alt}(\varphi)$*, where* $\mathsf{OP} \in \{\neg, \mathsf{X}, \flat\}$*;*
- $\mathsf{alt}(\varphi_1 \mathsf{OP} \varphi_2) \triangleq \mathsf{max}(\mathsf{alt}(\varphi_1), \mathsf{alt}(\varphi_2))$ *where* $\mathsf{OP} \in \{\vee, \mathsf{U}\}$*;*
- $\mathsf{alt}(\wp\varphi) \triangleq \mathsf{max}(\mathsf{alt}(\varphi), \mathsf{alt}(\wp))$ *where* $\wp = \wp_1 \ldots \wp_{|\wp|-1}$ *is a quantification prefix and* $\mathsf{alt}(\wp) \triangleq \sum_{i=1}^{|\wp|-1} \mathsf{switch}(\wp_i, \wp_{i+1})$*, where* $\mathsf{switch}(Q, Q') = 0$ *if* Q *and* Q' *are either both universal or both existential quantifiers, and* 1 *otherwise* [6]*.*

Another important fragment of SL is SL[1G] (here "1G" stands for One-Goal). Intuitively, SL[1G] is the fragment of GRADEDSL[NG] in which quantification is immediately followed

[5] In [MMPV14] the alternation number is described for all formulas of SL. Here, we only define it for GRADEDSL[NG] since this is enough for our purposes.

[6] Observe that formulas of the form $\wp\varphi$ have no free variables and thus one cannot form formulas of the form $\wp'\wp\varphi$.

by binding. The importance of this fragment stems from the fact that it strictly includes ATL* while maintaining the same complexity for both the model checking and the satisfiability problems, *i.e.* 2EXPTIME-COMPLETE [FAGV12, MMPV14]. However, it is commonly believed that Nash Equilibrium cannot be expressed in this fragment. Similarly, we give the following definition of GRADEDSL[1G]:

Definition 4.2.12 GRADEDSL[1G] *formulas are built inductively using the following grammar, with* $p \in \mathrm{AP}$, $\wp \in \mathrm{Qn}(\mathrm{V})$ *(*$\mathrm{V} \subseteq \mathrm{Vr}$*), and* $\flat \in \mathrm{Bn}$*:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \wp\flat\varphi,$$

where $\wp$ *is a quantification prefix over* $\mathsf{free}(\flat\varphi)$.

Recall (see [MMPV14]) that ATL* can be viewed as the fragment of SL[1G] in which occurrences of $\wp\flat$ are of the form

$$\langle\!\langle x_1 \rangle\!\rangle \cdots \langle\!\langle x_m \rangle\!\rangle [\![x_{m+1}]\!] \cdots [\![x_n]\!] (\beta_1, x_1)(\beta_2, x_2) \cdots (\beta_n, x_n),$$

where $\{\beta_1, \cdots, \beta_n\} = \mathrm{Ag}$ for $n = |\mathrm{Ag}|$, and $x_i \neq x_j$ for all $i \neq j$. In the usual notation of ATL*, the operator $\wp\flat$ is written $\langle\langle A \rangle\rangle$ where $A = \{\beta_1, \cdots, \beta_m\}$. In a similar way, but using tuples of strategies, we define a graded extension of ATL* as a fragment of GRADED$_\mathbb{N}$SL[1G]:

Definition 4.2.13 GRADEDATL* *formulas are built inductively using the following grammar, with* $p \in \mathrm{AP}$, $\wp$ *is of the form* $\langle\!\langle x_1, \cdots, x_m \rangle\!\rangle^{\geq g} [\![x_{m+1}, \cdots, x_n]\!]$ *where* $n = |\mathrm{Ag}|$, $m \leq n$, $g \in \mathbb{N} \cup \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$ *and* $x_i \neq x_j$ *for all* $i \neq j$*, and* $\flat$ *is of the form* $(\beta_1, x_1)(\beta_2, x_2) \cdots (\beta_n, x_n)$ *where* $\{\beta_1, \cdots, \beta_n\} = \mathrm{Ag}$*:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \wp\flat\varphi.$$

Finally, an important fragment of GRADEDSL (in which one can express uniqueness of strategy profiles) is when all grades are in $\mathbb{N}$.

Definition 4.2.14 *Let* GRADED$_\mathbb{N}$SL*,* GRADED$_\mathbb{N}$SL[NG] GRADED$_\mathbb{N}$SL[1G]*, and* GRADED$_\mathbb{N}$ATL* *denote the corresponding fragments in which all grades are from the set* $\mathbb{N}$.

4.3 Model-checking GRADEDSL

In this section we study the model-checking problem for GRADEDSL and show that it is decidable with a time-complexity that is non-elementary (i.e., not bounded by any fixed tower of exponentials). However, it is elementary if the number of blocks of quantifiers is fixed.

Definition 4.3.1 *The* model-checking problem for GRADEDSL (respectively, GRADEDSL[NG]) *is the following decision problem: given a formula* φ *from* GRADEDSL *(respectively,* GRADEDSL[NG]*) over some finite sets of atoms* AP*, agents* Ag*, and variables* Vr*, and given an arena* $\mathcal{A}$ *over the sets* AP *and* Ag*, decide whether* $\mathcal{A} \models \varphi$

When measuring computational complexity, the grades in formulas are written in unary.

For the algorithmic procedures, we follow an *automata-theoretic approach* [KVW00], reducing the decision problem for the logic to the emptiness problem of an automaton. The procedure we propose here extends that used for SL in [MMPV14]. The only case that is different is the new graded quantifier over tuples of strategies, i.e., we show how to convert a GRADEDSL formula φ into an automaton that accepts exactly the (tree encodings) of the assignments that satisfy φ.

Tree Automata. A Σ*-labeled* Υ*-tree* T is a pair $\langle T, V \rangle$ where $T \subseteq \Upsilon^+$ is prefix-closed (i.e., if $t \in T$ and $s \in \Upsilon^+$ is a prefix of t then also $s \in T$), and $V : T \to \Sigma$ is a labeling function. Note that every word $w \in \Upsilon^+ \cup \Upsilon^\omega$ with the property that every prefix of w is in T, can be thought of as a path in T. Infinite paths are called *branches*. *Nondeterministic tree automata* (NTA) are a generalization to infinite trees of the classical automata on words [Tho90]. *Alternating tree automata* (ATA) are a further generalization of nondeterministic tree automata [EJ91]. Intuitively, on visiting a node of the input tree, while an NTA sends exactly one copy of itself to each of the successors of the node, an ATA can send several copies to the same successor. We use the parity acceptance condition [KVW00].

For a set X, let $\mathtt{B}^+(X)$ be the set of positive Boolean formulas over X, including the constants **true** and **false**. A set $Y \subseteq X$ satisfies a formula $\theta \in \mathtt{B}^+(X)$, written $Y \models \theta$, if assigning **true** to elements in Y and **false** to elements in $X \setminus Y$ makes θ true.

Definition 4.3.2 *An Alternating Parity Tree-Automaton (*APT*) is a tuple* $\mathcal{M} \triangleq \langle \Sigma, \Delta, \mathrm{Q}, \delta, q_{\mathrm{o}}, \mathrm{F} \rangle$*, where*

- Σ *is the* input alphabet*,*
- Δ *is a set of* directions*,*
- Q *is a finite set of* states*,*
- $q_{\mathrm{o}} \in \mathrm{Q}$ *is an* initial state*,*
- $\delta : \mathrm{Q} \times \Sigma \to \mathtt{B}^+(\Delta \times \mathrm{Q})$ *is an* alternating transition function*, and*
- F*, an* acceptance condition*, is of the form* $(\mathrm{F}_1, \ldots, \mathrm{F}_k) \in (2^{\mathrm{Q}})^+$ *where* $\mathrm{F}_1 \subseteq \mathrm{F}_2 \ldots \subseteq \mathrm{F}_k = \mathrm{Q}$*.*

The set $\Delta \times \mathrm{Q}$ is called the set of *moves*. An NTA is an ATA in which each conjunction in the transition function δ has exactly one move (d, q) associated with each direction d.

An *input tree* for an APT is a Σ-labeled Δ-tree $\mathsf{T} = \langle \mathrm{T}, \mathsf{v} \rangle$. A *run* of an APT on an input tree $\mathsf{T} = \langle \mathrm{T}, \mathsf{v} \rangle$ is a $(\Delta \times \mathrm{Q})$-tree R such that, for all nodes $x \in \mathrm{R}$, where $x = (d_1, q_1) \dots (d_n, q_n)$ (for some $n \in \mathbb{N}$), it holds that *(i)* $y \triangleq (d_1, \dots, d_n) \in \mathrm{T}$ and *(ii)* there is a set of moves $\mathrm{S} \subseteq \Delta \times \mathrm{Q}$ with $\mathrm{S} \models \delta(q_n, \mathsf{v}(y))$ such that $x \cdot (d, q) \in \mathrm{R}$ for all $(d, q) \in \mathrm{S}$.

The acceptance condition allows us to say when a run is successful. Let R be a run of an APT $\mathcal{M}$ on an input tree T and $u \in (\Delta \times \mathrm{Q})^\omega$ one of its branches. Let $\mathsf{inf}(u) \subseteq \mathrm{Q}$ denote the set of states that occur in infinitely many moves of u. Say that u *satisfies the parity acceptance condition* $\mathrm{F} = (\mathrm{F}_1, \dots, \mathrm{F}_k)$ if the least index $i \in [1, k]$ for which $\mathsf{inf}(u) \cap \mathrm{F}_i \neq \emptyset$ is even. A run is *successful* if all its branches satisfy the parity acceptance condition F. An APT *accepts* an input tree T iff there exists a successful run R of $\mathcal{M}$ on T.

The *language* $\mathrm{L}(\mathcal{M})$ of the APT $\mathcal{M}$ is the set of trees T accepted by $\mathcal{M}$. Two automata are *equivalent* if they have the same language. The *emptiness problem* for alternating parity tree-automata is to decide, given $\mathcal{M}$, whether $\mathrm{L}(\mathcal{M}) = \emptyset$. The *universality problem* is to decide whether $\mathcal{M}$ accepts all trees.

4.3.1 From Logic to Automata

We reduce the model-checking problem of GRADEDSL to the emptiness problem for alternating parity tree automata [MMPV14]. The main step is to translate every GRADEDSL formula φ (i.e., φ may have free placeholders), arena $\mathcal{A}$, and state s, into an APT that accepts a tree if and only if the tree encodes an assignment χ such that $\mathcal{A}, \chi, s \models \varphi$.

We first describe the encoding, following [MMPV14]. Informally, the arena $\mathcal{A}$ is encoded by its "tree-unwinding starting from s" whose nodes represent histories, i.e., the St-labeled St-tree $\mathsf{T} \triangleq \langle \mathrm{Hst}(s), \mathsf{u} \rangle$ such that $\mathsf{u}(h)$ is the last symbol of h. Then, every strategy $\chi(e)$ with $e \in \mathsf{free}(\varphi)$ is encoded as an Ac-labelled tree over the unwinding. The unwinding and these strategies $\chi(e)$ are viewed as a single $(\mathrm{VAL} \times \mathrm{St})$-labeled tree where $\mathrm{VAL} \triangleq \mathsf{free}(\varphi) \to \mathrm{Ac}$.

Definition 4.3.3 *The* encoding of χ (w.r.t. $\varphi, \mathcal{A}, s$) *is the* $(\mathrm{VAL} \times \mathrm{St})$*-labeled* St*-tree* $\mathsf{T} \triangleq \langle \mathrm{T}, \mathsf{u} \rangle$ *such that* T *is the set of histories* h *of* $\mathcal{A}$ *starting with* s *and* $\mathsf{u}(h) \triangleq (\mathsf{f}, q)$ *where* q *is the last symbol in* h *and* $\mathsf{f} : \mathsf{free}(\varphi) \to \mathrm{Ac}$ *is defined by* $\mathsf{f}(e) \triangleq \chi(e)(h)$ *for all* $e \in \mathsf{free}(\varphi)$.[7]

We now state and prove a lemma that says one can translate every formula in GRADEDSL into an APT. It is proved by induction on the structure of the formula φ, as in [MMPV14]. The idea for handling the new case, i.e., the graded quantifier $\langle\!\langle x_1, \dots, x_n \rangle\!\rangle^{\geq g} \psi$, is to build

[7] In case $\mathsf{free}(\varphi) = \emptyset$, then f is the (unique) empty function. In this case, the encoding of every χ may be viewed as the tree-unwinding from s.

an APT that is a projection of an APT that itself checks that each of the g tuples of strategies satisfies ψ and that each pair of g tuples is distinct. The case that $g \in \mathbb{N}$ directly builds the required automaton (as is done for SL [MMPV14]), while the case that $g \in \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$ goes through logic. Write MSOL for monadic second-order logic in the signature of trees. We use the following two results that show how to express counting quantifiers in MSOL. The first is due to Rabin [Rab69] and is nicely explained in [Tho90].

Theorem 4.3.1 (MSOL and Automata) *For every MSOL formula $\alpha(Y)$ there exists an* APT *accepting the set of trees Y such that $\alpha(Y)$ holds. Conversely, for every* APT *there is an MSOL formula $\alpha(Y)$ that holds on those Y that are accepted by the* APT.

The second is due to [BKR10] and is proved using the composition technique.

Theorem 4.3.2 *For every MSOL formula $\alpha(\overline{X}, Y)$ and $\kappa \in \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$ there exists an MSOL formula $\beta(\overline{X})$ equivalent to "there exist κ many trees Y such that $\alpha(\overline{X}, Y)$".*

Lemma 4.3.1 *For every* GRADEDSL *formula φ, arena $\mathcal{A}$, and state $s \in \mathrm{St}$, there exists an* APT $\mathcal{M}$ *such that for all assignments χ, if T is the encoding of χ (w.r.t. $\varphi, \mathcal{A}, s$), then $\mathcal{A}, \chi, s \models \varphi$ iff $\mathsf{T} \in \mathrm{L}(\mathcal{M})$.*

Proof. As in [MMPV14] we induct on the structure of the formula φ to construct the corresponding automaton $\mathcal{M}$. The Boolean operations are easily dealt with using the fact that disjunction corresponds to non-determinism, and negation corresponds to dualising the automaton. The temporal operators are dealt with by following the unique play (determined by the given assignment) and verifying the required subformulas, e.g., for $\mathsf{X}\psi$ the automaton, after taking one step along the play, launches a copy of the automaton for ψ. All of these operations incur a linear blowup in the size of the automaton. The only case that differs from SL is the quantification, i.e., we need to handle the case that $\varphi = \langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}\psi$. Recall that $\mathcal{G}, \chi, s \models \langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}\psi$ iff there exists g many tuples $\overline{\sigma_i}$ of strategies such that: $\overline{\sigma_a} \neq \overline{\sigma_b}$ for $a \neq b$, and $\mathcal{G}, \chi[\overline{x} \mapsto \overline{\sigma_i}], s \models \psi$ for $0 \leq i < g$.

There are two cases.

Case $g \in \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$. Consider $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}\psi$. By induction, there is an APT $\mathcal{D}$ for ψ. Apply Theorem 4.3.1 to translate $\mathcal{D}$ into an MSOL formula α, then apply Theorem 4.3.2 to get an MSOL formula β that holds iff "there exist g many tuples of trees such that α" (recall that a tuple of trees is coded as a single tree). Finally, apply Theorem 4.3.1 to convert β into the required APT.

Note that the blowup in the translations (MSOL to APT, and closure under "there exists κ many trees") is non-elementary.

Case $g \in \mathbb{N}$. Let $\mathcal{M}$ be the APT for ψ, given by induction. We show how to build an NPT for φ that mimics the definition of φ: it will be a projection of an APT, which itself is the

intersection of two automata, one checking that each of the g tuples of strategies satisfies ψ, and the other checking that each pair of the g tuples of strategies is distinct.

In more detail, introduce a set of fresh variables $X \triangleq \{x_i^j : i \leq n, j \leq g\}$, and consider the formulas ψ^j (for $j \leq g$) formed from ψ by renaming x_i (for $i \leq n$) to x_i^j. Define $\psi' \triangleq \wedge_{j \leq g} \psi^j$. Note that, by induction, each ψ^j has a corresponding APT and thus, there is an APT $\mathcal{B}_{\mathcal{M},X}$ for ψ' (conjunction can be dealt using universal-choice). Note that the input alphabet for $\mathcal{B}_{\mathcal{M},X}$ is $(\mathsf{free}(\psi') \to \mathrm{Ac}) \times \mathrm{St}$ and that $X \subseteq \mathsf{free}(\psi')$.

On the other hand, let $\mathcal{C}_X$ be an APT with input alphabet $(\mathsf{free}(\psi') \to \mathrm{Ac}) \times \mathrm{St}$ that accepts a tree $\mathsf{T} = \langle \mathrm{T}, \mathsf{v} \rangle$ if and only if for every $a \neq b \leq g$ there exists $i \leq n$ and $h \in \mathrm{T}$ such that $\mathsf{v}(h) = (f, q)$ (where q is the last symbol of h) and $f(x_i^a) \neq f(x_i^b)$. To build $\mathcal{C}_X$ simply form the conjunction of automata $\mathcal{C}_X^{a,b}$ (for $a \neq b \leq g$), each of which is the disjunction of automata $\mathcal{C}_X^{a,b,i}$ (for $i \leq n$) that checks (by nondeterministically guessing a path) that there exists a history h starting in s such that $f(x_i^a) \neq f(x_i^b)$ (where the first co-ordinate of $\mathsf{v}(h)$ is f).

Form the APT $\mathcal{D}_{\mathcal{M},X}$ for the intersection of $\mathcal{B}_{\mathcal{M},X}$ and $\mathcal{C}_X$ (formed using universal-choice).

Now, using the classic transformation [MS95], we remove alternation from the APT $\mathcal{D}_{\mathcal{M},X}$ to get an equivalent NPT $\mathcal{N}$ (note that this step costs an exponential). Finally, use the fact that NPTs are closed under projection (with no blowup) to get an NPT for the language $proj_X(L(\mathcal{N}))$ of trees that encode assignments χ satisfying φ.

For completeness we recall this last step. If L is a language of Σ-labeled trees with $\Sigma \triangleq \mathrm{A} \to \mathrm{B}$, and $X \subset \mathrm{A}$, then the X-*projection of* L, written $proj_{\mathrm{X}}(L)$, is the language of Σ'-labeled trees with $\Sigma' \triangleq \mathrm{A} \setminus \mathrm{X} \to \mathrm{B}$ such that $\mathsf{T} \triangleq \langle \mathrm{T}, \mathsf{v} \rangle \in proj_{\mathrm{X}}(L)$ if and only if there exists an X-labeled tree $\langle \mathrm{T}, \mathsf{w} \rangle$ such that the language L contains the tree $\langle \mathrm{T}, \mathsf{u} \rangle$ where $\mathsf{u} : \mathrm{T} \to (\mathrm{A} \to \mathrm{B})$ maps $t \in \mathrm{T}$ to $\mathsf{v}(t) \cup \mathsf{w}(t)$. Now, if $\mathcal{N}$ is an NPT with input alphabet $\Sigma \triangleq \mathrm{A} \to \mathrm{B}$, and if $\mathrm{X} \subset \mathrm{A}$, then there is an NPT with input alphabet $\Sigma' \triangleq \mathrm{A} \setminus \mathrm{X} \to \mathrm{B}$ with language $proj_{\mathrm{X}}(L(\mathcal{N}))$.

The proof that the construction is correct is immediate. □

4.3.2 Decidability and Complexity of Model Checking

We can immediately conclude that model-checking graded strategy logic is decidable.

Theorem 4.3.3 *The model-checking problem for* GRADEDSL *is decidable. Moreover, the complexity is not bounded by any fixed tower of exponentials.*

Proof. For decidability, use Lemma 4.3.1 to transform the arena and φ into an APT and test its emptiness. The lower bound already holds for SL [MMPV14]. □

In the rest of this section we supply a finer analysis of the complexity of various fragments of graded strategy logic. To do this, we first analyse the number of states of the APT

constructed in Lemma 4.3.1.

All the cases in the induction incur at most a linear blowup except for the quantification case. For the quantification case, in case $g \in \{\aleph_0, \aleph_1, 2^{\aleph_0}\}$ the blowup is non-elementary.

In case $g \in \mathbb{N}$ then the translation incurs an exponential blowup. Indeed, the number of states of the APT $\mathcal{B}_{\mathcal{M},X}$ is $g \times n$ times the number of states of the APT for ψ, and since $\mathcal{C}_X$ consists of the conjunction of $g(g-1)$ automata (one for each pair of tuples), and each such automaton has $O(n)$ many states, the number of states of $\mathcal{C}_X$ is $O(ng^2)$. Thus, the number of states of the APT $\mathcal{D}_{\mathcal{M},X}$ is polynomial in the number of states of the APT for ψ. Finally, the translation from an APT to an NPT results in an exponentially larger automaton [KVW00].

In case all grades are from $\mathbb{N}$ and the formulas are written using the universal-strategy quantifier shorthand, we can easily modify the construction to handle quantifier-blocks in one shot as if they were a single quantifier, i.e., with a single exponential blowup. For instance, suppose $\phi = \langle\langle y_1, \ldots, y_m \rangle\rangle^{\geq h} \langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g} \psi$ (additional quantifiers are treated similarly). As in the proof, let $\mathcal{M}$ be the APT for ψ and take $\mathcal{D}_{\mathcal{M},X}$. Now, instead of immediately removing alternation and projecting, build $\mathcal{D}_{\mathcal{M}',Y}$ where $\mathcal{M}'$ is $\mathcal{D}_{\mathcal{M},X}$ and $Y \triangleq \{y_i^j : i \leq m, j \leq h\}$. Finally, remove alternation from $\mathcal{D}_{\mathcal{M}',Y}$ to get an NPT $\mathcal{N}'$, and then apply the $(X \cup Y)$-projection to the language of $\mathcal{N}'$ to get the desired APT for ϕ. Note that the size of $\mathcal{D}_{\mathcal{M}',Y}$ is exponential in the number of states of $\mathcal{M}$ since the costly step of removing alternation is performed only once. Similarly, to deal with a block of universal strategy quantifiers simply use dualisation. For instance, to deal with $\phi = [\![y_1, \ldots, y_m]\!]^{<h} [\![x_1, \ldots, x_n]\!]^{<g} \psi$ apply the previous procedure to the equivalent formula $\neg\langle\langle y_1, \ldots, y_m \rangle\rangle^{\geq h} \langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g} \neg\psi$ (recall that negating an APT is done by dualisation, which incurs no blowup).

Recall that GRADED$_{\mathbb{N}}$SL denotes the fragment of GRADEDSL in which all grades are restricted to $\mathbb{N}$.

Theorem 4.3.4

1. *For every $k \geq 1$, the model-checking problem for* GRADED$_{\mathbb{N}}$SL *formulas of quantifier-block rank at most k is in* $(k+1)$EXPTIME.

2. *For every $k \geq 1$, the model-checking problem for* GRADED$_{\mathbb{N}}$SL *formulas of the form $\varphi = \wp\psi$, where $\wp$ is a quantifier-block of φ and ψ is of quantifier-block rank at most $k-1$, is in* kEXPTIME.

3. *The model-checking problem for* GRADED$_{\mathbb{N}}$SL *formulas of the form $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g} \psi$ or $[\![x_1, \ldots, x_n]\!]^{<g} \psi$ is in* EXPTIME *w.r.t. the parameter g (written in unary).*

Proof. Before proving the upper bounds, recall that the complexity of checking emptiness (resp. universality) of an APT is in EXPTIME in the number of states [KVW00].

For item 1 proceed as follows. As discussed after Lemma 4.3.1, for the case that all grades are in $\mathbb{N}$, the number of states of the APT for φ is a tower of exponentials whose height is the quantifier-block rank of φ. This gives the $(k+1)$EXPTIME bound.

For item 2 suppose that $\varphi = \wp\psi$ where $\wp$ consists of, say, n existential quantifiers (resp. universal quantifiers). The quantifier-block rank of ψ is $k-1$. Moreover, the APT $\mathcal{D}_\psi$, whose number of states is non-elementary in $k-1$, has the property that it is non-empty (resp. universal) if and only if the arena satisfies $\wp\psi$. Conclude that model checking $\wp\psi$ can be solved in kEXPTIME.

For item 3 first observe that the size of the APT constructed in Lemma 4.3.1 grows quadratically in g. The statement follows by recalling that the complexity of model-checking formulas of this form is exponential in the number of states of the APT. □

Theorem 4.3.5 *For every $k \geq 1$, the model-checking problem for* GRADED$_{\mathbb{N}}$SL[NG] *formulas of alternation number at most k is in* $(k+1)$EXPTIME *and* kEXPSPACE*-hard.*

Proof. The lower bound already holds for SL[NG] [MMPV14]. For the upper bound, as for SL[NG], note that principal formulas are "state formulas", i.e., their truth value only depends on the state in which they are interpreted (this is because they have no free placeholders). Thus, one can apply the following marking algorithm to a formula Φ. For every principal subformula $\varphi = \wp\psi$ of Φ for which ψ is quantifier-free, and for every state s of $\mathcal{A}$, introduce a new atomic proposition p_φ. Label s by p_φ (i.e., extend the labelling function at s to include p_φ) iff $\mathcal{A}, s \models \varphi$. Replace the subformula φ of Φ by the atom p_φ, and repeat this process. Observe that the complexity of marking a state is in $(k+1)$EXPTIME where $k = \mathsf{alt}(\varphi)$ (indeed, $\mathsf{alt}(\varphi) = \mathsf{qbr}(\varphi)$ since $\varphi = \wp\psi$ and ψ is quantifier-free, and thus one can apply Theorem 4.3.4 item 1). Also, the cost of the whole marking algorithm is the sum of the costs of all the marking rounds, and the number of rounds is at most the size of the formula. After the marking algorithm has completed, one is left with a Boolean combination of atomic propositions which is trivial to evaluate at each state. Thus the total time is at most $(k+1)$EXPTIME. □

Theorem 4.3.6 *The model-checking problem for* GRADED$_{\mathbb{N}}$SL[1G] *is* 2EXPTIME-COMPLETE.

Proof. The lower-bound already holds for SL[1G] [MMPV14]. The upper-bound follows the same algorithm as for SL[1G] [MMPV14] but uses the (no more complex) construction from Lemma 4.3.1 for the strategic quantifier $\langle\langle x_1, \ldots, x_n \rangle\rangle^{\geq g}$ instead of $\langle\langle x \rangle\rangle$. □

Finally, since model checking ATL* is 2EXPTIME-HARD, and ATL* is a syntactic fragment of GRADED$_{\mathbb{N}}$ATL*, which itself is a syntactic fragment of GRADED$_{\mathbb{N}}$SL[1G], we have that:

Theorem 4.3.7 *The model-checking problem for* GRADED$_{\mathbb{N}}$ATL* *is* 2EXPTIME-COMPLETE.

4.4 Analysing Games using GRADEDSL

In this section we describe how to use the models and formulas of GRADEDSL to reason about solution concepts from game theory. In particular, we show how to use arenas to model games of finite or infinite duration, and GRADEDSL to express the uniqueness of winning strategies, Nash equilibria, subgame-perfect equilibria, and Pareto-efficient profiles. We prove that deciding the uniqueness for all these solution concepts is not more expensive than merely deciding their existence, i.e., 2EXPTIME. We note that while the problem of deciding the existence of, e.g., NE, is 2EXPTIME-hard [GHW17], it is not known whether deciding the uniqueness of NE is as hard. This intriguing question remains open, and we reserve it for future work.

4.4.1 Strategic Form and Infinitely Repeated Games

The *Strategic Form* is the most familiar representation of strategic interactions in Game Theory. A game written in this way amounts to a representation of every player's preference for every state of the world, in the special case where states of the world depend only on the players' combined actions.

Definition 4.4.1 *A* strategic form game *is a tuple* $(N, A, (\preceq_i)_{i\in N})$*, where:*

- N *is a finite set of* n *players, indexed by* i*;*
- $A = A_1 \times \ldots \times A_n$*, where* A_i *is a finite set of actions available to player* i*. Each vector* $a = (a_1, \ldots, a_n) \in A$ *is called an action profile;*
- *each* $\preceq_i$ *is a total pre-order (i.e., reflexive and transitive) on* A*.*

Note that a common way to give the preference relation is by using a *payoff function* $pay : A \to \mathbb{R}$, which assigns a real number to every element in A. In this case, the preference relation $\preceq_i$ is defined by having $a \preceq_i a'$ iff $pay(a) \leq pay(a')$.

A classic way to model players that repeatedly interact with each other in a game $(N, A, (\preceq_i)_{i\in N})$ is by *infinitely repeated games*, see e.g., [OR94]. We will illustrate by formalising an iterated prisoner's dilemma (Section 4.4.3).

4.4.2 Quasi-Quantitative Games and Objective-LTL Games

As expected, we can also specify games by arenas and a payoff function on plays. In this section we define quasi-quantitative games, a generalisation of *objective*-LTL *games* [KPV14].

	C	D
C	$-1, -1$	$-4, 0$
D	$0, -4$	$-3, -3$

Figure 4.1: Prisoner's Dilemma in Strategic Form. Each row corresponds to a possible action for player 1, each column corresponds to a possible action for player 2, and each cell corresponds to one possible outcome. Payoffs of the players for an outcome are written in the corresponding cell, with the payoff of player 1 listed first.

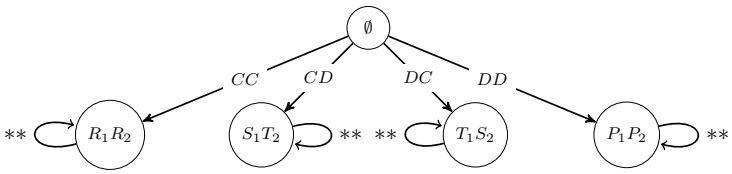


Figure 4.2: Arena of the Prisoner's dilemma.

Let $\mathcal{A}$ be an arena with n agents. Let $m \in \mathbb{N}$, for each agent $\alpha_i \in \mathrm{Ag}$, a *quasi-quantitative objective* is a tuple $S_i \triangleq \langle \mathsf{f}_i, L_i^1, \ldots, L_i^m \rangle$, where $\mathsf{f}_i : \{0,1\}^m \to \mathbb{Z}$, and each L_i^j is a set of sequences of sets of atomic propositions. If π is an infinite path, then agent α_i receives payoff $\mathsf{f}_i(b_i^\pi) \in \mathbb{N}$ where the j'th bit of b_i^π is 1 if and only if $\mathsf{ap}(\pi) \in L_i^j$. We assume agents are trying to maximise their payoffs. The tuple $\mathcal{G} = \langle \mathcal{A}, S_1, \ldots, S_n \rangle$ is called a *quasi-quantitative game*. In case each $f_i : \{0,1\}^m \to \{-1, 1\}$ we say that the game is *win/lose*. If $\sum_{1 \leq i \leq n} \mathsf{f}_i(b_i^\pi) = 0$ for all π, then $\mathcal{G}$ is *zero-sum*, otherwise it is a *non zero-sum*.

In case each L_i^j is the set of models of an LTL formula φ_i^j over AP, we call $\mathcal{G}$ an *objective*-LTL *game*. We introduce the following useful shorthand. For $\alpha_i \in \mathrm{Ag}$ and $a \in \{0,1\}^m$, define η_i^a for the LTL formula $\bigwedge_{j \leq m} \psi_{a,j}$ where $\psi_{a,j} = \varphi_i^j$ if $a_j = 1$ and $\psi_{a,j} = \neg\varphi_i^j$ if $a_j = 0$.

4.4.3 Example: The Prisoner's Dilemma (PD)

A natural way to draw games in strategic form is via an n-dimensional matrix. Figure 4.1 contains the matrix of the classic Prisoner's Dilemma. The two actions are C (co-operate) and D (defect). The payoffs are the prison sentences (in years) that the prisoners get for each pair of actions that they choose.[8] The dilemma faced by the prisoners is that, whatever the

[8]The story behind the dilemma is this: Two people have been arrested for robbing a bank and placed in separate isolation cells. Each has two possible choices, remaining silent (action C) or confessing (action D). If a robber confesses and the other remains silent, the former is released and the latter stays in prison for a long time.

choice of the other prisoner, each is better off playing action D. But the result obtained when both play action D is worse than if they both play action C. Observe that the actual numbers are not important for there to be a dilemma. Rather, it is enough that the induced preference relation is $\mathbf{T}_i > \mathbf{R}_i > \mathbf{P}_i > \mathbf{S}_i$, where $\mathbf{R}_i$ represents the *reward* that player i receives if both cooperate; $\mathbf{P}_i$ is the *punishment* that player i receives if both defect; $\mathbf{T}_i$ is the *temptation* that player i receives as a sole defector, and $\mathbf{S}_i$ is the *sucker* payoff that player i receives as a sole cooperator.

We can model the Prisoner's Dilemma with the arena in Figure 4.2. Agent α_i (representing player i) has objective $S_i \triangleq \langle \mathsf{f}_i, \varphi_i^1, \varphi_i^2, \varphi_i^3, \varphi_i^4 \rangle$ where $\varphi_i^1 \triangleq \mathsf{X}\,\mathbf{S}_i$, $\varphi_i^2 \triangleq \mathsf{X}\,\mathbf{P}_i$, $\varphi_i^3 \triangleq \mathsf{X}\,\mathbf{R}_i$, and $\varphi_i^4 \triangleq \mathsf{X}\,\mathbf{T}_i$ and f_i returns the value of its input vector interpreted as a binary number, *e.g.*, $\mathsf{f}_i(0100) = 4$ that represents the payoff in which φ_i^3 is true. In words, we have two agents α_1 and α_2. Each agent has two actions, C and D. For each possible pair of moves, the game goes in a state whose atomic propositions represent the preferences.

It is well known that in the Prisoner's Dilemma the only Nash equilibrium is for both players to defect. The reason is that each prisoner must hedge against the possibility of the other one defecting. However, it is clear that if they would have both cooperated, they would be better off. If there was a way for one prisoner to later punish a defection of the other prisoner, it may not have to hedge, and would be able to cooperate instead. Such behaviours emerge as a rational choice, for example, when one considers the infinitely repeated prisoner's dilemma, in which the prisoners repeat the basic strategic form game infinitely often.[9] Indeed, it is well known that for this iterated game (for example, with a payoff that is the mean-payoff of the prison sentences [Bin92]), a new Nash equilibrium emerges, in which both players use the so called *Grim* strategy, in which a prisoner cooperates as long as the other prisoner cooperates, but switches to always defect the first time the other prisoner defects. Observe that the resulting infinite play of this Nash equilibrium has both players cooperating all the time. The core reason that the pair of grim strategies is a Nash equilibrium for the mean-payoff version of the iterated prisoner's dilemma is that this payoff ignores the price of being a 'sucker' on any finite prefix of the play, i.e., that the mean-payoff of a play is independent of any finite prefix of that play — other properties of the mean are not needed, and constitute unimportant noise. Indeed, the same Nash equilibria would emerge if, for example, one takes instead of the mean-payoff the maximal payoff that repeats infinitely often.

More generally, given a game in strategic form, and a preference relation $\preceq_i$ over its set of possible outcomes A, one can define a new preference relation $\preceq_i^\infty$ over A^ω by assigning a payoff to every subset of A and assigning to each play the set $inf(\pi) \subseteq A$ of outcomes in A

If both confess they are both convicted, but will get early parole. If both remain silent, they get a lighter sentence (e.g., on firearms possession charges).

[9] Alternatively, one can introduce the threat of a punishment for defecting by considering a probabilistic version in which it is unclear to the prisoners how many repetitions will be used. Note, however, that a fixed number of repetitions turns out to be essentially the same as playing only once [Bin92].

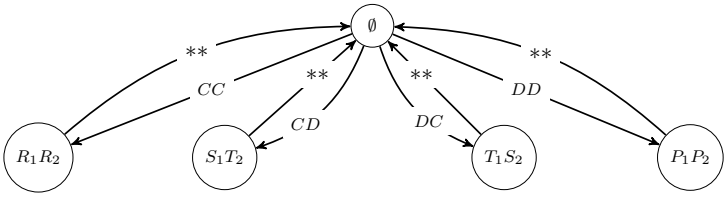


Figure 4.3: Arena of the Iterated Prisoner's dilemma.

that appear infinitely often in π. Formally, let $F_i : 2^A \to \mathbb{Z}$ be a function mapping subsets of A to integers, and define $\pi \preceq_i^\infty \pi'$ iff $F_i(inf(\pi)) \leq F_i(inf(\pi'))$.[10] For example, for the iterated prisoners' dilemma, setting $F_i(X)$ to be the number of elements y in A such that $y \prec_i x$ where x is a $\preceq_i$-maximal element of X, results in a game with the same set of Nash equilibria as in the mean payoff version.

We formalise the infinitely repeated prisoner's dilemma as an objective-LTL game. The arena is in Figure 4.3. The preferences, for agent α_i, are defined by the objective $S_i \triangleq \langle \mathsf{f}_i, \varphi_i^1, \varphi_i^2, \varphi_i^3, \varphi_i^4 \rangle$ where $\varphi_i^1 \triangleq \mathsf{G}\,\mathsf{F}\,\mathbf{S}_i$, $\varphi_i^2 \triangleq \mathsf{G}\,\mathsf{F}\,\mathbf{P}_i$, $\varphi_i^3 \triangleq \mathsf{G}\,\mathsf{F}\,\mathbf{R}_i$, and $\varphi_i^4 \triangleq \mathsf{G}\,\mathsf{F}\,\mathbf{T}_i$, and f_i as before.

4.4.4 Illustrating GRADEDSL: uniqueness of solutions

We illustrate how to express with GRADEDSL some important solution concepts in Game Theory. We start with the concept of winning strategy that is useful in zero-sum games, and then we analyse the well known solution concepts, such as Nash and subgame-perfect equilibria, that are used in non zero-sum games. We use ordinary SL quantifiers (i.e., $\langle\!\langle x \rangle\!\rangle$, $[\![x]\!]$) since, as observed in Section 4.2, these are expressible in GRADEDSL.

4.4.4.1 Winning strategies

In two-agent win-lose zero-sum games the main solution concept is the *winning strategy*. That is, if G is such an objective-LTL game, then a strategy for agent α_1 is *winning* if and only if for all strategies of agent α_2, the resulting induced play has payoff 1 for agent α_1. This can be expressed in SL as follows:

$$\phi_{WS}(x) \triangleq [\![y]\!](\alpha_1, x)(\alpha_2, y) \bigvee_{f_1(a)=1} \eta_1^a$$

where η_1^a is the LTL formula defined in Section 4.4.2. Thus, the following formula expresses that there is a unique winning strategy for agent α_1:

$$\langle\!\langle x \rangle\!\rangle^{\geq 1} \phi_{WS}(x) \wedge \neg \langle\!\langle x \rangle\!\rangle^{\geq 2} \phi_{WS}(x) \tag{4.1}$$

[10] This is reminiscent of the Muller acceptance condition in automata theory.

Observe that this is a formula of GRADED$_{\mathbb{N}}$SL[NG] of alternation number 1. Thus, by Theorem 4.3.4 we get:

Theorem 4.4.1 *Deciding if a given agent in a two-agent zero-sum objective-*LTL *game has a unique winning strategy can be solved in* 2EXPTIME.

We illustrate with an example. In [MMS15] the authors describe a two-agent game named "Cop and the Robber", played in a maze, in which the objective of the Robber is to reach an exit (and thus the objective of the Cop is to ensure the Robber never reaches the exit). The authors describe two closely related mazes in which the Robber has, respectively, exactly one and exactly two winning strategies. Both these properties can easily be expressed by GRADEDSL. For instance, the Robber has a single LTL objective $\mathsf{F}\, exit$, and the following formula of GRADED$_{\mathbb{N}}$SL expresses that the Robber has exactly one winning strategy:

$$\langle\!\langle x \rangle\!\rangle^{\geq 1} [\![y]\!] (\text{Robber}, x)(\text{Cop}, y)\, \mathsf{F}\, exit \wedge \neg \langle\!\langle x \rangle\!\rangle^{\geq 2} [\![y]\!] (\text{Robber}, x)(\text{Cop}, y)\, \mathsf{F}\, exit.$$

4.4.4.2 Nash Equilibria

The central solution concept in non zero-sum games is the Nash Equilibrium. A tuple of strategies, one for each agent, is called a *strategy profile*. A strategy profile is a *Nash equilibrium (NE)* if no agent can increase his payoff by unilaterally choosing a different strategy. A game may have zero, one, or many NE.

Consider the case that each agent α_i has a general objective tuple $S_i \triangleq \langle \mathsf{f}_i, \varphi_i^1, \ldots, \varphi_i^m \rangle$. Recall the definition of the LTL formulas η_i^a from Section 4.4.2.

For $\overline{x} \triangleq (x_1 \ldots x_n)$ and $\overline{y} \triangleq (y_1 \ldots y_n)$, the following formula says that if all agents follow $\overline{x}$, then no agent i gets a better payoff by deviating and following y_i:

$$\phi_{DEV}(\overline{x}, \overline{y}) \triangleq \bigwedge_{i=1}^{n} \bigwedge_{a \in \{0,1\}^m} (\flat(\overline{x}/y_i)\eta_i^a) \rightarrow \bigvee_{\mathsf{f}_i(a') \geq \mathsf{f}_i(a)} \flat(\overline{x})\eta_i^{a'}$$

where $\flat(\overline{x}) = (\alpha_1, x_1) \ldots (\alpha_n, x_n)$, and

$$\flat(\overline{x}/y_i) = (\alpha_1, x_1) \ldots (\alpha_{i-1}, x_{i-1})(\alpha_i, y_i)(\alpha_{i+1}, x_{i+1}) \ldots (\alpha_n, x_n).$$

Then, the following SL formula says that $\overline{x}$ is a NE

$$\phi_{NE}(\overline{x}) \triangleq [\![y_1]\!] \ldots [\![y_n]\!] \phi_{DEV}(\overline{x}, \overline{y}),$$

and the following GRADED$_{\mathbb{N}}$SL[NG] formula expresses that there is a unique NE:

$$\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq 1} [\![y_1]\!] \ldots [\![y_n]\!] \phi_{DEV}(\overline{x}, \overline{y}) \wedge \neg \langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq 2} [\![y_1]\!] \ldots [\![y_n]\!] \phi_{DEV}(\overline{x}, \overline{y})$$

Thus, by Theorem 4.3.4 we get:

Theorem 4.4.2 *Deciding if an objective-*LTL *game has a unique NE can be solved in* 2EXPTIME.

Rational synthesis can be formalised as the problem of deciding if a given game has a NE such that the resulting play satisfies a given LTL formula Ψ. In our setting, we get the following result by replacing ϕ_{DEV} by $\Psi \wedge \phi_{DEV}$ in the previous formula:

Theorem 4.4.3 *Deciding if an objective-*LTL *game has a unique NE satisfying an* LTL *formula* Ψ *can be solved in* 2EXPTIME.

4.4.4.3 Pareto efficiency

A strategy profile is said to be *Pareto efficient (PE)* if there is no other strategy profile that makes some agent better off without making another agent worse off. The formula $\phi_{PE}(\overline{x}) \triangleq [\![x'_1]\!] \dots [\![x'_n]\!]\psi(\overline{x}, \overline{x'})$ expresses that $\overline{x}$ is PE where $\psi(\overline{x}, \overline{x'})$ is

$$\bigwedge_{i \leq n} \bigwedge_{(a,a') \in X_i} \Big((\flat(\overline{x})\eta_i^a \wedge \flat(\overline{x'})\eta_i^{a'}) \rightarrow \bigvee_{j \neq i} \bigvee_{(c,c') \in Y_i} (\flat(\overline{x})\eta_j^c \wedge \flat(\overline{x'})\eta_j^{c'}) \Big),$$

where $(a, a') \in X_i$ iff $\mathsf{f}_i(a') > \mathsf{f}_i(a)$, where $(c, c') \in Y_i$ iff $\mathsf{f}_j(c') < \mathsf{f}_j(c)$, $\flat(\overline{x}) \triangleq (\alpha_1, x_1) \dots (\alpha_n, x_n)$, and $\flat(\overline{x'}) \triangleq (\alpha_1, x'_1) \dots (\alpha_n, x'_n)$. Using graded modalities, we can thus express that there is a unique PE using the following $\text{GRADED}_{\mathbb{N}}\text{SL}[\text{NG}]$ formula of alternation number 1:

$$\langle\!\langle x_1, \dots, x_n \rangle\!\rangle^{\geq 1} \phi_{PE}(\overline{x}) \wedge \neg \langle\!\langle x_1, \dots, x_n \rangle\!\rangle^{\geq 2} \phi_{PE}(\overline{x}).$$

Thus, by using Theorem 4.3.4 we get:

Theorem 4.4.4 *Deciding if an objective-*LTL *game has a unique Pareto efficient profile can be solved in* 2EXPTIME.

4.4.4.4 Subgame-perfect equilibria

Finally, we end with a discussion of the problem of deciding if a game has a unique subgame-perfect equilibrium, and give an upper bound. It has been argued (in [Umm06, KPV14]) that NE may be implausible when used for sequential games (of which infinitely repeating games are central examples), and that a more robust notion is subgame-perfect equilibrium [Sel65].

Informally, a strategy profile is a subgame-perfect equilibrium if it is a NE in every reachable subgame. Here is the mathematical definition instantiated for quasi-quantitative games (following the definition in [OR94] for extensive-form games). Given a history $h \in \text{Hst}(s_I)$ ending in state s, say $h = us$, and a strategy $\sigma \in \text{Str}(s_I)$, the *h-translation* of σ is the strategy $\sigma_{|h} \in \text{Str}(s)$ that maps $h' \in \text{Hst}(s)$ to $\sigma(u \cdot h')$. Given a quasi-quantitative

game $\mathcal{G} = \langle \mathcal{A}, S_1, \ldots, S_n \rangle$, the profile $\sigma_1, \ldots, \sigma_n$ is a *subgame-perfect equilibrium (SPE)* iff for all histories $h \in \mathrm{Hst}(s_I)$, the profile $\sigma_{1|h}, \ldots, \sigma_{n|h}$ is a NE in $\mathcal{G} = \langle \mathcal{A}_{|h}, S_{1|h}, \ldots, S_{n|h} \rangle$ where $\mathcal{A}_{|h}$ is the same arena as $\mathcal{A}$ but with s as the initial state, and if $S = \langle \mathsf{f}_i, L_i^1, \ldots, L_i^m \rangle$ then $S_{|h} = \langle \mathsf{f}_i, H_i^1, \ldots, H_i^m \rangle$ where $\pi \in H_i^j$ iff $u \cdot \pi \in L_i^j$. The point is that the payoff in $\mathcal{G}_{|h}$ applies to the whole path (i.e., starting from s_I), even though the strategies only apply after h.

Using the notation in the previous paragraph, suppose each L_i^j is prefix-independent, i.e., $\pi \in L_i^j$ iff $\pi_{\geq n} \in L_i^j$ for all $n \geq 1$ (here $\pi_{\geq n}$ is the suffix of π starting at position n). In this case, $H_i^j = L_i^j$. Observe that the assumption that the objectives are prefix-independent is not too restrictive. Indeed, as discussed in Section 4.4.3, in many infinitely repeated games the outcome ignores all finite prefixes of the play.

Thus, suppose $\mathcal{G}$ is an objective-LTL game in which the set of models of each φ_i^j is prefix-independent. The following formula of SL expresses that $\overline{x}$ is an SPE:[11]

$$\phi_S(\overline{x}) \triangleq [\![z_1]\!] \ldots [\![z_n]\!][\![y_1]\!] \ldots [\![y_n]\!](\alpha_1, z_1) \ldots (\alpha_n, z_n)\, \mathsf{G}\, \phi_{DEV}(\overline{x}, \overline{y}).$$

Indeed, since $[\![\cdot]\!]$ commutes with G, the formula $\phi_S(\overline{x})$ is equivalent to

$$[\![z_1]\!] \ldots [\![z_n]\!](\alpha_1, z_1) \ldots (\alpha_n, z_n)\, \mathsf{G}\, \phi_{NE}(\overline{x}),$$

which is true in $\mathcal{A}, \chi, s_I$ iff for all histories h starting in s_I and ending, say, in state s, we have that $\mathcal{A}, \chi', s \models \phi_{NE}$ where the strategy $\chi'(x)$ is the h-translation of the strategy $\chi(x)$, i.e., the profile $\chi'(x_1), \ldots, \chi'(x_n)$ is a NE in $\mathcal{G}_{|h}$.

Using graded modalities, we can thus express there is a unique SPE (assuming each φ_i^j is prefix-independent) as the following $\textsc{Graded}_{\mathbb{N}}\mathrm{SL}[\mathrm{NG}]$ formula of alternation number 1:

$$\langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq 1} \phi_S(\overline{x}) \wedge \neg \langle\!\langle x_1, \ldots, x_n \rangle\!\rangle^{\geq 2} \phi_S(\overline{x}).$$

Thus, by using Theorem 4.3.4 we get:

Theorem 4.4.5 *Deciding if an objective-*LTL *game with prefix-independent objectives φ_i^j has a unique SPE can be solved in* 2ExpTime.

4.5 Conclusion

The Nash equilibrium is the foundational solution concept in game theory. The last twenty years have witnessed the introduction of many logical formalisms for modeling and reasoning about solution concepts, and NE in particular [MMV10a, MMPV14, CHP10, BLLM09, LLM10, Bel15, GHW17]. These formalisms are useful for addressing qualitative questions such as *"does the game admit a Nash equilibrium?"*. Among others, Strategy Logic (SL)

[11] Previous formalisations of SPE overlook the need for a condition like prefix-independence [Umm06, MMPV14, KPV14].

has come to the fore as a general formalism that can express and solve this question, for LTL objectives, in 2EXPTIME. Contrast this with the fact that this question is 2EXPTIME-complete even for two player zero-sum LTL games [PR89].

One of the most important questions about NE in computational game theory is *"does the game admit more than one NE?"* [PC79, CHS99] This issue has been deeply investigated in game theory and is shown to be very challenging [SLCB13, ZG11, Pav12, AKH02, ORS93, BBV86, Fra92, GK93]. Prior to this work, no logic-based technique, as far as we know, solved the corresponding decision problem, i.e., whether or not a given game has a unique NE.[12]

In this chapter we introduced GRADEDSL to address and solve the unique NE problem. We have demonstrated that GRADEDSL is elegant, simple, and very powerful, and can solve the unique NE problem for LTL objectives in 2EXPTIME, and thus at the same complexity that is required to merely decide if a NE exists. We also illustrated that one can express the uniqueness of other solution concepts, including winning strategies, subgame-perfect equilibria, and Pareto-efficient profiles, all in 2EXPTIME. We remark that the exact complexity of the existence of a unique NE, and of counting the number of NE, has been studied for other representations of games, notably games in strategic form with rational co-efficients and considering mixed strategies [GZ89, CS08]. The exact complexity of the existence of NE in our representation is currently unknown. Finally, our work gives the first algorithmic solution to the model-checking problem of a graded variant of ATL*, and proves it to be 2EXPTIME-COMPLETE.

In the multi-agent setting, reasoning about epistemic alternatives plays a key role. Thus, an important extension would be to combine the knowledge operators in SLK [ČLMM14] with the graded quantifiers we introduced for GRADEDSL. Since strategic reasoning under imperfect information has an undecidable model-checking problem [DT11], one may restrict to memoryless strategies as was done for SLK. More involved, would be to add grades to the knowledge operators, thus being able to express "there exists at least g equivalent worlds" [vdHM92].

Finally, another direction is to implement GRADEDSL and its model-checking procedure in a formal verification tool. A reasonable approach would be, for example, to extend the tool SLK-MCMAS [ČLMM14, ČLM15].

[12] In the related work section we discussed the logic GSL [MMMS15, MMMS17] that, although motivated by the need to address the unique NE problem, only supplies a model-checking algorithm for a very small fragment of GSL that cannot express LTL goals and, it is assumed, is not able to express the existence of NE.

Part III

Strategies and their complexity

CHAPTER 5

Reasoning about Natural Strategic Ability

Contents

5.1 Introduction

Logics for strategic reasoning provide powerful tools to reason about multi-agent systems [AHK02, vdHW02, Sch04, JvdH04, CHP10, MMPV14]. The logics allow to express properties of agents' behavior and its dynamics, driven by their individual and collective goals. An important factor here is interaction between the agents, which can be cooperative as well as adversarial. Specifications in agent logics can be then used as input to *model checking* [CE81, QS82], which makes it possible to verify the correct behavior of a multi-agent system using recently developed practical automatic tools [LQR09, ČLMM14, ČLM15].

A fundamental contribution in this field is *Alternating-Time Temporal Logic (*ATL**)* and its fragment ATL [AHK02]. ATL* formulas are usually interpreted over *concurrent game structures (CGS)* which are labeled state-transition systems that model synchronous interaction among agents. For example, given a CGS modeling a system with k agents and a shared resource, the ATL formula $\langle\langle A\rangle\rangle \mathsf{Fgrant}$ expresses the fact that the set of agents A can ensure that, regardless of the actions of the other agents, an access to the resource will be eventually granted. The specification holds if agents in A have a collective strategy whose every execution path satisfies Fgrant. As in game theory, strategies are understood as conditional plans, and play a central role in reasoning about purposeful agents.

Formally, strategies in ATL* (as well as in other logics of strategic reasoning, such as Strategy Logic [CHP10, MMPV14]) 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. That makes sense from a mathematical point of view, and also in case we think of strategic ability of a machine (robot, computer program). We claim, however, that the approach is not very realistic for reasoning about human behavior. This is because humans are very bad at handling combinatorially complex objects. A human strategy should be relatively simple and "intuitive" or "natural" in order for the person to understand it, memorize it, and execute it. This applies even more if the human agent has to come up with the strategy on its own.

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

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

In the former group, [AW09] investigates strategic properties of agents with bounded memory, while [ALNR09, ALNR10, BF10a, BF10b] extend temporal and strategic logics to handle agents with bounded resources. Issues related to bounded rationality are also investigated in [BCS08, HO09, GSW14].

The latter category is much richer and includes extensions of ATL* with explicit reasoning about actions and strategies [vdHJW05, Ågo06, WvdHW07, HLMW14], and logics that combine features of temporal and dynamic logic [HK82, NJ09]. A variant of STIT logic that enables reasoning about strategies and their performance in the object language [DB16]. Also, plans in agent-oriented programming are in fact rule-based descriptions of strategies. In particular, reasoning about agent programs using strategic logics was investigated in [BFVW06, ADLM07, ALDM08, DJ10, YS12].

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

5.2 A Logic for Natural Ability

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

Example 5.2.1 (Motivating examples) *The main application domain that we have in mind is* reasoning about usability. *Consider, e.g., a ticket vending machine at a railway station. Intuitively, it is not enough that a customer has a strategy to successfully buy the right ticket. If the strategy is too complex, most people will be unable to follow it, and the machine will be practically useless.*

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

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

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

5.2.1 Syntax

Alternating-time temporal logic (ATL, for short) [AHK02] generalizes branching-time temporal logic CTL* by replacing path quantifiers E, A with *strategic modality* $\langle\langle A\rangle\rangle$. Informally, $\langle\langle A\rangle\rangle\gamma$ reads "there exists a strategy for the coalition A such that, no matter how the other players will act, the formula γ is satisfied. *Natural* ATL (NatATL, for short) is obtained by replacing in ATL the modality $\langle\langle A\rangle\rangle$ with the bounded strategic modality $\langle\langle A\rangle\rangle^{\leq k}$. Intuitively, $\langle\langle A\rangle\rangle^{\leq k}\gamma$ reads as coalition A has a collective strategy of *size less or equal than* k to enforce the property γ. As for ATL, the formulas of NatATL make use of classical temporal operators: "X" ("in the next state"), "G" ("always from now on"), "F" ("now or sometime in the future"), U (strong "until"), and W (weak "until").

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

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

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

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.

5.2.2 Concurrent Game Structures

The semantics of NatATL is defined over concurrent game structures[AHK02].

Definition 5.2.1 (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* $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_{|\mathbb{A}\mathrm{gt}|}:\}$*, states* St*, actions* Act*, atomic propositions* $Prop$*, and a propositional valuation* $V : St \to 2^{Prop}$*. The function* $d : \mathbb{A}\mathrm{gt} \times St \to 2^{Act}$ *defines availability of actions. The (deterministic) transition function* t *assigns a successor state* $q' = t(q, \alpha_1, \ldots, \alpha_{|\mathbb{A}\mathrm{gt}|})$ *to each state* $q \in St$ *and any tuple of actions* $\alpha_i \in d(a_i, q)$ *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 choice of group* A *at state* q *by* $d_A(q) = \prod_{a_i \in A} d_{a_i}(q)$*.*

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

A *path* $\lambda = q_0 q_1 q_2 \ldots$ in a CGS is an infinite sequence of states such that there is a transition between each q_i, q_{i+1}. $\lambda[i]$ denotes the ith position on λ, $\lambda[i, j]$ the part of λ between positions i and j, and $\lambda[i, \infty]$ the suffix of λ starting with i. We denote with Λ the

set of all paths. Similarly, a *history* $h = q_0 q_1 q_2 \dots q_n$ 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 $H = St^+$ the set of all the histories in the model.

5.2.3 Strategies and Their Complexity

To properly interpret NatATL formulas, we introduce the concept of *natural strategies* and their *outcomes* over a CGS. Following Schobbens [Sch04], we distinguish between strategies *with* and *without* the recall of the hitherto history of the game. We will use R to refer to the semantics of strategic ability arising for strategies with recall, and r for strategies without recall. In this section, we show how natural strategies without recall can be defined. The other kind of strategies is proposed and studied in Section 5.4.

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

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

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

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

Example 5.2.2 *Consider the following* r*-strategy* s*:*

1. $(\neg\mathsf{ticket} \wedge \neg\mathsf{selected},\ select)$*;*

2. $(\neg\mathsf{ticket} \wedge \mathsf{selected},\ pay)$;

3. $(\top,\ idle)$.

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

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^{r}_A. The "outcome" function $out(q, s_A)$ returns the set of all paths that occur when agents A execute strategy s_A from state q onward. Formally, given a state $q \in St$, a subset of agents A and a collective memoryless strategy s_A, we define:

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

5.2.4 Semantics of NatATL

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

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

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

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

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

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

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

Example 5.2.3 *When designing a game, the designer can define the game level by the complexity of the smallest winning strategy for the player. Using* NatATL*, we can say that the level of game G is k iff $G \models_{\mathrm{r}} \langle\langle a\rangle\rangle^{\leq k}\mathsf{Fwin} \wedge \neg\langle\langle a\rangle\rangle^{\leq k-1}\mathsf{Fwin}$.*

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

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

5.3 Model Checking for Natural Memoryless Strategies

In this section we show how to solve the model checking problem for NatATL with r-strategies, i.e. NatATL$_{\mathrm{r}}$. We start with the simpler case in which the bound of the strategies is given as a constant and prove that the model checking problem is polynomial in the size of the game structure. Then, we consider the case in which the bound k is a variable and prove that the model checking problem becomes $\mathbf{\Delta_2^P} - complete$. Regarding this latter case, we also investigate the setting in which NatATL$_{\mathrm{r}}$ formulas have only one strategic operator, i.e. 1NatATL$_{\mathrm{r}}$, and show that the model checking problem turns out to be $\mathbf{NP} - complete$. The results and the proofs presented in this section have been inspired by [Sch04, JD06].

5.3.1 Model Checking for Small Strategies

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

Theorem 5.3.1 *The model checking problem for* NatATL$_{\mathrm{r}}$ *with fixed k is in* $\mathbf{P}$.

Proof. First, consider the formula $\varphi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, in which $A \subseteq \mathbb{A}\mathrm{gt}$ 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 $compl_{\Sigma}(s_A) \leq k$. Thus, we have $O(|Prop|^k)$ possible kinds of guarded actions and so $O((|Prop|^k)^k) = O(|Prop|^{k^2})$ possible lists. Given the collective strategy s_A, we can prune the CGS by removing all edges that disagree with s_A. This operation costs, in the worst case, $O(|t|)$, where t is the transition relation of the input CGS. So far we have solved the strategic operator of the input formula φ and we are left with a structure S that can be seen as a Kripke structure. Now, we can reduce our problem to model checking the CTL formula $\mathsf{A}\gamma$ ("for all paths γ") over S by using the standard model checking algorithm for CTL [CE81], well-known to have complexity $O(|t| \cdot |\gamma|)$. The total complexity is thus $O(|Prop|^{k^2} \cdot (|t| + (|t| \cdot |\gamma|))) = O(|Prop|^{k^2} \cdot |t| \cdot |\gamma|)$, and hence polynomial in the size of the model.

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

5.3.2 Model Checking: General Case

We now study the complexity for NatATL$_{\mathrm{r}}$ with the bound of the strategic modalities given as variables. We consider two different cases: formulas with a single strategic operator followed

by a simple temporal subformula, and formulas with possibly nested strategic operators. For the former case we show an **NP** procedure, and by a reduction from SAT that the problem is **NP**-complete. For the latter case we show a $\mathbf{\Delta_2^P}$ procedure and by a reduction from SNSAT the $\mathbf{\Delta_2^P}$-completeness.

Theorem 5.3.2 *Model checking* 1NatATL$_\mathrm{r}$ *is in* **NP**.

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. By assumption, we can use strategies with no a priori bounded size. To overcome this, to construct a collective strategy s_A we use an oracle that returns a collective strategy for A. We can now conclude by using the same reasoning done in the proof of Theorem 5.3.1. In particular, since we use an oracle over a polynomial algorithm the overall complexity is **NP**. □

We continue by showing a matching lower bound by means of a reduction from the well-known SAT problem. We first provide the reduction and then show that it is correct in Theorem 5.3.3. 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 an initial state q_0, a state for each clause C_i in φ, a state for each literal in C_i and the state $q_\top$. The set of $Prop$ is $\{\mathsf{C_1}, \ldots, \mathsf{C_n}, \mathsf{x_1}, \ldots, \mathsf{x_m}, win\}$. Furthermore, we label each state clause/variable with its proposition and $q_\top$ with win. The flow of the game is defined as follows. The refuter decides at the beginning of the game which clause C_i will have to be satisfied: it is done by proceeding from the initial state q_0 to a clause state q_i. At q_i, verifier decides (by proceeding to a proposition state $q_{i,j}$) which of the literals $x_j^{s(i,j)}$ from C_i will be attempted. Finally, at $q_{i,j}$, verifier attempts to prove C_i by declaring the underlying propositional variable x_j true (action $\top$) or false (action $\bot$). If $\mathbf{v}$ succeeds (i.e., if it executes $\top$ for x_j^+, or executes $\bot$ for x_j^-), then the system proceeds to the winning state $q_\top$. Otherwise, the system stays in $q_{i,j}$. It is important to note that by definitions of $Prop$ and V, we know that $\mathbf{v}$ can use just one action (i.e. truth value) for each variable. This is due to the fact that we use as strategies the guarded actions that are determined directly from the atomic proposition instead from states.

More formally, $M_\varphi = \langle \mathbb{A}\mathrm{gt}, St, Act, d, t, Prop, V \rangle$, 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 $St_{prop} = \{q_{1,1}, \ldots, q_{1,m}, \ldots, q_{n,1}, \ldots, q_{n,m}\}$;

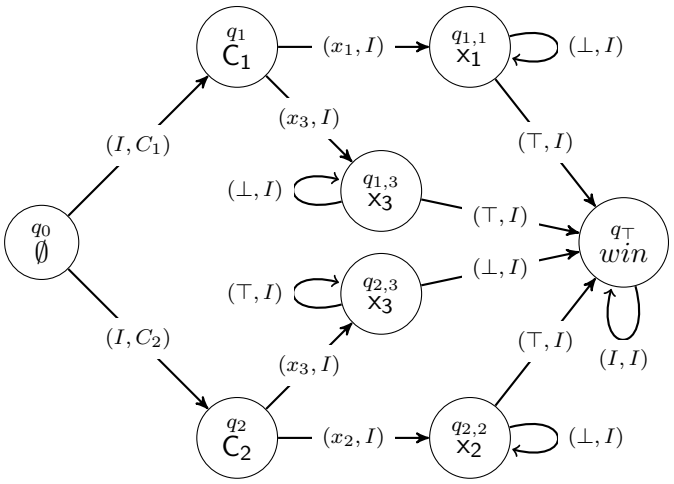


Figure 5.1: A CGS for checking satisfiability of $\varphi = (x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$. Action I denotes "idle." For simplicity, we omit the states that have no incoming edges.

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

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

As an example, model M_φ for $\varphi = (x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$ is presented in Figure 5.1.

Theorem 5.3.3 *$SAT(n, m, \varphi)$ iff $M_\varphi, q_0 \models \langle\langle \mathbf{v} \rangle\rangle^{\leq n+m} F\mathsf{win}$*

Proof. ($\Rightarrow$) Firstly, if there is a valuation 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: (i) 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 (ii) 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. ($\Leftarrow$) Conversely, if $M_\varphi, q_0 \models \langle\langle \mathbf{v} \rangle\rangle^{\leq n+m} F\mathsf{win}$, then there is a strategy $s_\mathbf{v}$ such that $q_\top$ is achieved for all paths from $out(q_0, s_\mathbf{v})$. But then the valuation, which assigns propositions $x_1, \ldots, x_m$ with the same values as $s_\mathbf{v}$, satisfies φ. □

By Theorem 5.3.2 and Theorem 5.3.3, the following result holds.

Corollary 5.3.1 *Model checking* 1NatATL$_\mathrm{r}$ *is* **NP***-complete.*

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

Theorem 5.3.4 *Model checking* NatATL$_{\mathrm{r}}$ *is in* $\mathbf{\Delta_2^P}$.

Proof. We make use of a bottom-up procedure based on the one introduced in the proof of Theorem 5.3.1. Precisely, take an arbitrary formula φ of NatATL$_{\mathrm{r}}$ and consider its inner part that is of the kind $\psi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, with γ being a formula over boolean connectives and atomic propositions. Now, apply over ψ the procedure used in the proof of Theorem 5.3.2 that we know to be $\mathbf{NP}$. Once ψ is solved, use the same $\mathbf{NP}$ procedure to solve ψ', a formula that contains ψ and a strategic operator, and so on for each strategic operator in φ. This means that we use an oracle over a polynomial procedure for each strategic operator in φ. Summing up, the total complexity to solve a formula in NatATL$_{\mathrm{r}}$ is $\mathbf{P^{NP}} = \mathbf{\Delta_2^P}$. □

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

Definition 5.3.1 *Given a fixed number r and $1 \leq i \leq r$, a SNSAT instance is defined as follows:*

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

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

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

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

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

Given such an instance of SNSAT, we construct a sequence of concurrent game structures M_i in a similar way to the construction used for the reduction from SAT. That is, clauses and variables $x_{k,i}$ are handled in exactly the same way as before. Moreover, if z_h, with $1 \leq h < i$, occurs as a positive literal in φ_i, we embed M_h in M_i, and add a transition to the initial state q_0^h of M_h. If $\neg z_h$ occurs in φ_i, we do almost the same: the only difference is that we split the transition into two steps, with a state neg_h^i (labeled with a proposition neg) added in between. More formally, $M_i = \langle \mathbb{A}\mathrm{gt}, St^i, Act^i, d^i, t^i, Prop^i, V^i \rangle$, 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 $St_{neg} = \{neg_1^i, \ldots, neg_{i-1}^i\}$;
- $Act^i = \{I, C_1, \ldots, C_n, x_1, \ldots, x_m, z_1, \ldots, z_r, \top, \bot\}$;
- $d^i(v, q_0^i) = d^i(v, q_\top) = d^i(v, neg_h^i) = \{I\}$, $d^i(v, q_j^i) = \{x_k \mid x_{k,i}$ or $\neg x_{k,i}$ is in $C_j^i\} \cup \{z_h \mid z_h$ or $\neg z_h$ is in $C_j^i\}$, $d^i(v, q_{j,k}^i) = \{\top, \bot\}$; $d^i(\mathbf{r}, q_0^i) = \{C_1, \ldots, C_n\}$ and $d^i(\mathbf{r}, q) = \{I\}$ with $q \in St \setminus \{q_0^i\}$. For $q \in St^{i-1}$, we simply include the function from M_{i-1}: $d^i(a, q) = d^{i-1}(a, q)$;
- $t^i(q_0^i, I, C_j) = q_j^i$, $t^i(q_j^i, x_k, I) = q_{j,k}^i$, $t^i(q_j^i, z_h, I) = q_0^h$ if $s^i(j, h) = +$ and neg_h^i otherwise, $t^i(neg_h^i, I, I) = q_0^h$, $t(q_{j,k}^i, \top, I) = q_\top$ if $s^i(j, k) = +$, and $q_{j,k}^i$ otherwise, $t(q_{j,k}^i, \bot, I) = q_\top$ if $s^i(j, k) = -$, and $q_{i,j}$ otherwise. For $q \in St^{i-1}$, we include the transition function from M_{i-1}: $t^i(q, \alpha_1, \alpha_2) = t^{i-1}(q, \alpha_1, \alpha_2)$;
- $Prop^i = \{\mathsf{C}_1^i, \ldots, \mathsf{C}_n^i, \mathsf{x}_1^i, \ldots, \mathsf{x}_m^i, \mathsf{win}, \mathsf{neg}\}$;
- $V(q_0^i) = \emptyset$, $V(q_j^i) = \mathsf{C}_j^i$, $V(q_{j,k}^i) = \mathsf{x}_k^i$, $V(neg_h^i) = \mathsf{neg}$ and $V(q_\top) = \mathsf{win}$.

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

As an example, model M_3 is presented in Figure 5.2.

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

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

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

Lemma 5.3.1 $\forall 1 \leq l \leq i \leq r$*:* $M_l, q_0^l \models \phi_i$ *iff* $M_l, q_0^l \models \phi_{i-1}$.

The proof of the lemma is a straightforward adaptation of [JD06, Lemma 5].

Theorem 5.3.5 $\forall 1 \leq i \leq r$ *:* z_i *is true iff* $M_i, q_0^i \models \phi_i$

Proof. Induction on i:

(i) For $i = 1$, we use the proof of Theorem 5.3.3.

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

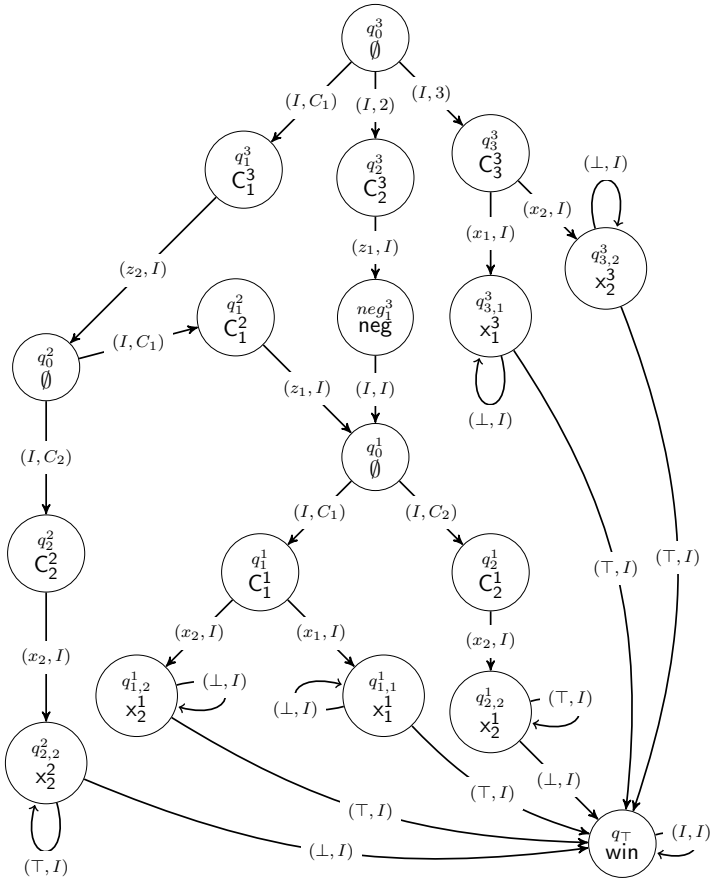


Figure 5.2: A CGS for $\varphi_3 = z_2 \wedge \neg z_1 \wedge (x_1 \vee x_2)$, $\varphi_2 = z_1 \wedge \neg x_2$, and $\varphi_1 = (x_1 \vee x_2) \wedge \neg x_2$. For simplicity, we omit the states that have no ingoing edges.

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

($\Rightarrow$) Firstly, if z_i is true then there is a valuation v of X_i that makes φ_i true. We construct s_v as in the proof of Theorem 5.3.3. In case that some $x_{k,i}^s$ has been chosen in clause C_j^i then we define the guarded action $(\mathsf{C}_\mathsf{j}^\mathsf{i},\ x_k)$ and we are done. In case that some z_h^- has been chosen in clause C_j^i, where $h < i$, we have (by induction) that $M_h, q_0^h \models \neg\phi_h$. By Lemma 5.3.1, also $M_h, q_0^h \models \neg\phi_i$, and hence $M_i, q_0^h \models \neg\phi_i$. So we can make the same choice (i.e., we define the guarded action $(\mathsf{C}_\mathsf{j}^\mathsf{i},\ z_h)$) in s_v, and this will lead to state neq_h^i, in which it holds that $neg \wedge A\mathsf{X}\,\neg\phi_i$. In case that some z_h^+ has been chosen in clause C_j^i, we have that $M_h, q_0^h \models \phi_h$. By Lemma 5.3.1, also $M_h, q_0^h \models \phi_i$. That is, there is a strategy s'_v in M_h such that $(\neg\mathsf{neg})\,\mathsf{U}\,(\mathsf{win} \vee (\mathsf{neg} \wedge A\mathsf{X}\,\neg\phi_{i-1}))$ holds for all paths from $out(q_0^h, s'_v)$. Then, we can merge s'_v into s_v.

($\Leftarrow$) Conversely, if $M_i, q_0^i \models \phi_i$, then there is a strategy s_v that enforces $(\neg\mathsf{neg})\,\mathsf{U}\,(\mathsf{win} \vee (\mathsf{neg} \wedge A\mathsf{X}\,\neg\phi_{i-1}))$. First, we consider the clause C_j^i with guarded action $(\mathsf{C}_\mathsf{j}^\mathsf{i},\ x_k)$, i.e. for which a propositional state is chosen by s_v. The strategy defines a valuation for X_i that

satisfies these clauses. For the other clauses, i.e. there is a guarded action $(\mathsf{C}^{\mathsf{i}}_{\mathsf{j}},\ z_h)$, we have two possibilities:

- s_v chooses q_0^h in the state corresponding to C_j^i. Neither win nor neg have been encountered on this path yet, so we can take s_v to demonstrate that $M_i, q_0^h \models \phi_i$, and hence $M_h, q_0^h \models \phi_i$. By Lemma 5.3.1, also $M_h, q_0^h \models \phi_h$. By induction, z_h must be true, and hence clause C_j^i is satisfied.
- s_v chooses neg_h^i in the state corresponding to C_j^i. Then, it must be that $M_i, neg_h^i \models A\mathsf{X}\,\neg\phi_{i-1}$, and hence $M_h, q_0^h \models \neg\phi_{i-1}$. By Lemma 5.3.1, also $M_h, q_0^h \models \neg\phi_h$. By induction, z_h must be false, and hence clause C_j^i (containing $\neg z_h$) is also satisfied.

□

By Theorem 5.3.4 and Theorem 5.3.5, the following result holds.

Corollary 5.3.2 *Model checking* NatATL$_{\mathrm{r}}$ *is* $\mathbf{\Delta_2^P}$*-complete.*

5.4 A Logic for Natural Strategic Ability of Agents with Memory

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

5.4.1 Natural Recall

Let $Reg(L)$ be the set of regular expressions over the language L (with the standard constructors $\cdot, \cup, *$ representing concatenation, nondeterministic choice, and finite iteration). A *natural strategy with recall* (or R-strategy) s_a for agent a is a sequence of appropriate pairs from $Reg(\beta(2^{Prop})) \times Act$. That is, it consists of pairs $(r,\ \alpha)$ where r is a regular expression over $\beta(2^{Prop})$ and α is an action available in $last(h)$, i.e. $\alpha \in d_a(last(h))$, for all histories $h \in H$ consistent with r. Formally, given a regular expression r and the language $L(r)$ on words generated by r, a history $h = q_0 \dots q_n$ is consistent with r iff $\exists\, b \in L(r)$ such that $|h| = |b|$ and $\forall_{0 \le i \le n}\, h[i] \models b[i]$. Similarly to r-strategies, the last pair on the list is assumed to be simply $(\top^*,\ idle)$. The set of such strategies is denoted by Σ_a^{R}. Finally, $match(\lambda[0,i], s_a)$ is the smallest $n \le size(s_a)$ such that $\forall_{0 \le j \le i} \lambda[j] \models cond_n(s_a)[j]$ and $act_n(s_a) \in d_a(\lambda[i])$. A *collective natural strategy* for agents $A = \{a_1, \dots, a_{|A|} :\}$ is a tuple of individual natural strategies $s_A = (s_{a_1}, \dots, s_{a_{|A|}})$. The set of such strategies is denoted by Σ_A^{R}. Again, $out(q, s_A)$ returns the set of all paths of strategy s_A. For strategies with recall, we simply replace "$match(\lambda[i], s_a)$" with "$match(\lambda[0,i], s_a)$" in the definition from Section 5.2.3.

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

Example 5.4.1 *Consider the following* R*-strategy* s*:*

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

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

Remark 5.4.1 *Note that natural strategies with recall are by definition finite. Thus, they do not exactly correspond to the notion of* perfect recall *where an agent may specify different choices for each of the infinitely many finite histories of the game. In this sense, our representations are similar to* finite memory strategies *from [Ves13].*

5.4.2 NatATL for Strategies with Recall

Now it is easy to define the semantics of natural strategic ability for agents with recall. Formally, we construct the semantic relation $\models_{\mathrm{R}}$ by replacing "$\models_{\mathrm{r}}$" with "$\models_{\mathrm{R}}$" and Σ^{r}_A with Σ^{R}_A in the clauses from Section 5.2.4.

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

5.4.3 Relation to Natural Memoryless Strategies

It is well known that the semantics of ATL based on memoryless and perfect recall strategies coincide (under perfect information). This follows from the correctness of the model checking algorithm in [AHK02], cf. also [Sch04]. Precisely, there is a strategy with recall to enforce a given temporal property γ iff there is a memoryless strategy to enforce γ. We now prove that the same does not hold in NatATL.

Theorem 5.4.1 *The following results hold in* NatATL*:*

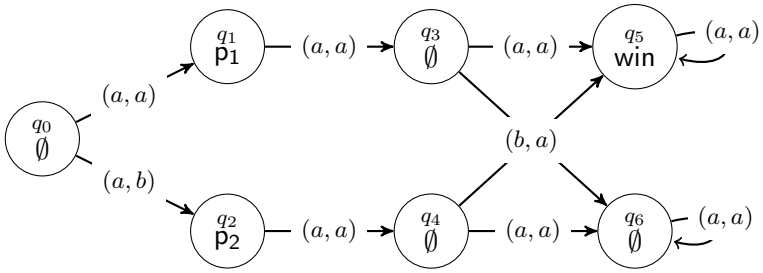


Figure 5.3: A counterexample for Theorem 5.4.1.

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

Proof. (1) Firstly, given an r-strategy s it is possible to construct an R-strategy s' that has the same behavior of s. In fact, for each guarded action $(\theta,\ \alpha)$ of s with $\theta \in \beta(2^{Prop})$ and $\alpha \in Act$ we can write a guarded action $(r,\ \alpha)$ in s' such that $r = (\top^*) \cdot \theta$.
(2) Consider, the CGS M in Figure 5.3, where there are two players 1 and 2. Each transition is labeled with a couple of actions (α, β), where α is an action of 1 and β is an action of 2. We show that the formula $\varphi = \langle\langle 1\rangle\rangle^{\leq k} F\mathsf{win}$ is true by using a natural strategy with recall and is false for each possible natural memoryless strategy. The following strategy with recall s satisfies φ:

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

To be convinced, first recall that natural memoryless strategies are defined only over atomic propositions, so, if there are states with the same set of atomic propositions, then there exists at most one guarded action and then at most one possible action in these states. Therefore, in the model in Figure 5.3 with a natural memoryless strategy it is impossible to define two different behaviors in the states q_3 and q_4, and for this reason player 1 has no memoryless strategies to reach a state labeled with win. □

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

the semantics of ATL defines memoryless strategies as functions from states to actions, which allows for arbitrary granularity.

To remove the limit of natural memoryless strategies, we define a subclass of models named *fully distinguishing models*. The idea behind this kind of models is the one used in [AHK02, LMO08] to define the distinguishing models. The formal definition follows.

Definition 5.4.1 *Given a CGS M, we say that M is a* fully distinguishing model *iff, for all $S \subseteq St$, there exists $\mathsf{p} \in Prop$ such that:*

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

Theorem 5.4.2 *Given a fully distinguishing model M, a state q, a subset of agents A, and a formula $\varphi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, it holds that: $M, q \models_{\mathrm{r}} \varphi$ iff $M, q \models_{\mathrm{R}} \varphi$.*

Proof. ($\Rightarrow$) For this direction we use the proof of Theorem 5.4.1(1).
($\Leftarrow$) Assume now that $M, q \models_{\mathrm{R}} \varphi$. By definition, there is a strategy $s_A \in \Sigma^{\mathrm{R}}_A$ such that $compl(s_A) \leq k$, and for each path $\lambda \in out(q, s_A)$, we have $M, \lambda \models_{\mathrm{R}} \gamma$. From s_A, let us construct a memoryless strategy $s'_A \in \Sigma^{\mathrm{r}}_A$ such that the following facts hold: (i) $\forall \lambda \in out(q, s'_A)$, we have $M, \lambda \models_{\mathrm{r}} \gamma$ and (ii) $compl(s'_A) \leq compl(s_A)$. We start at the state q. We know that M is a fully distinguishing model, so the state q is distinguishable with respect to the other states of M. Consider for simplicity that the only atomic proposition that is true in q is q. We fix $s'_A(\mathsf{q}) = s_A(\mathsf{q})$, where $s_A(\mathsf{q})$ represents the action in the strategy with recall s_A for the regular expression q that is just an atomic proposition. Consider now the successors of q consistent with $s_A(\mathsf{q})$. $\forall q' \in out(q, s_A(\mathsf{q}))$ we take the atomic proposition q' that is true just in q' and fix $s'_A(\mathsf{q}') = s_A(\mathsf{q} \cdot \mathsf{q}')$, where $\mathsf{q} \cdot \mathsf{q}'$ is the regular expression that is composed by the atomic propositions q and q' that are only true in q and q', respectively. We repeat this procedure until we get to a fixpoint, i.e. all states are covered, except possibly for some states that are unreachable when we execute s_A. By the definition, we also know that these states satisfy the guarded action $(\top,\ idle)$. To conclude the proof, we just need to show that (i) and (ii) hold. Item (i) can be proved by induction. For the lack of space, we omit the details. Item (ii) follows by the construction of s'_A. In fact, we construct s'_A from s_A, that is for each guarded action $(\mathsf{q},\ \alpha)$ of s'_A there is a guarded action $(r,\ \alpha)$ of s_A, where $r = r_0 \cdot \ldots \cdot r_n$ and $r_n = \mathsf{q}$, then $compl(s'_A) \leq compl(s_A)$. □

5.5 Model Checking for Natural Strategies with Recall

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

5.5.1 Model Checking for Small Strategies

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

Theorem 5.5.1 *The model checking problem for* NatATL_{R} *with fixed* k *is in* $\mathbf{\Delta_2^P}$.

Proof. Assume for the moment that we have a NatATL_{R} formula $\varphi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, where $A \subseteq \mathbb{A}\text{gt}$ and γ is a formula over boolean connectives and atomic propositions. As for the solution in NatATL_{r}, we know that the collective strategy we can assign to A, namely s_A, is bounded and, precisely, we have that $compl_{\Sigma}(s_A) \leq k$. The main difference between r-strategies and R-strategies regards the underlying domains, i.e., we move from boolean propositional formulas to regular expressions over boolean propositional formulas (both over atomic propositions). Recall that, regular expressions are a combination of atomic propositions ($Prop$), boolean connectives ($Bool$), and standard constructors (Con). Thus, in this case, we have $(|Prop + Bool + Con|)^k$ possible different guarded actions and $(|Prop + Bool + Con|^k)^k = |Prop + Bool + Con|^{k^2}$ possible lists. Given s_A, we cannot prune M since we have an R-strategy. Let us consider now the unwinding of M and remove all edges that are not in accordance with s_A. It is important to observe that the unwinding of a model can be infinite and thus we need to consider a bounded unwinding. A possibility would be to consider the tree unwinding with depth $|St| + 1$ as we are sure that after this bound there is a loop. Unfortunately, this is a too big upper-bound. Indeed, checking all paths of the unwinding, in the worst case (i.e. each state is connected with all states of the model), requires $|St|^{|St|}$ steps, that is exponential on the number of states. To avoid this exponential blow-up we use a guessing oracle. The algorithm that solves the model checking problem for NatATL_{R} with fixed k is depicted in Figure 5.4. The $mCheck^k_{\text{NatATL}_{\text{R}}}$ algorithm uses the oracle depicted in Figure 5.5. It guesses a history h of length $|St| + 1$ that satisfies the CTL formula $\neg\mathsf{A}\gamma$. If such a history exists then the oracle returns true and $mCheck^k_{\text{NatATL}_{\text{R}}}$ returns false because the original formula wants γ to hold. Conversely, if the oracle returns false and $mCheck^k_{\text{NatATL}_{\text{R}}}$ returns true then $M, q \models_{\text{R}} \varphi$. For the complexity, we use an oracle over a polynomial procedure. So, we have that the total complexity is $\mathbf{P^{NP}} = \mathbf{\Delta_2^P}$. To conclude the proof, let us drop the initial limitations on the formula. If the formula has more than one strategic operator, then we proceed as in the proof of Theorem 5.3.1 and so we use a bottom-up procedure, i.e. we first solve the formula with the inner most strategic operator, then we update the formula and repeat the procedure, until we reach the outermost formula. This requires to use a further oracle over a polynomial procedure that works over a polynomial procedure itself. Hence, we have that the overall complexity is $\mathbf{P^{P^{NP}}} = \mathbf{P^{NP}} = \mathbf{\Delta_2^P}$. □

```
1  Algorithm mCheck^k_{NatATL_R}(M, q, φ):
2  for every s_A with compl(s_A) ≤ k do
3  t = Oracle(M, q, φ, s_A)
4  return (¬t)
```

Figure 5.4: Model checking algorithm for NatATL_R with fixed k

```
1  Algorithm Oracle(M, q, φ, s_A):
2  Guess h ∈ H^{|St+1|}(q)
3  if h is inconsistent with s_A
4    return false
5  else
6    return mCheck_CTL(h, h[0], ¬Aγ)
```

Figure 5.5: Oracle

5.5.2 Model Checking: General Case

We now study the complexity for NatATL_R in case the bound over the strategies is not fixed. In particular, we study separately the cases in which the formula under exam has one or more nested strategic operators. For the former we show a $\mathbf{\Sigma_2^P}$ procedure and, for the latter, a $\mathbf{\Delta_3^P}$ one. As for the memoryless case, the proofs in this section have been inspired by [Sch04, JD06].

Theorem 5.2.2 *Model checking* 1NatATL_R *with variable* k *is in* $\mathbf{\Sigma_2^P}$.

Proof. Consider the formula $\langle\langle A\rangle\rangle^{\leq k}\gamma$, where $A = \{a\}$ and γ is a formula over boolean connectives and atomic propositions. By assumption, the bound of the strategy is not fixed. For this reason, to construct a strategy s_a we use an $\mathbf{NP}$ oracle that constructs a strategy for a. We report in Figure 5.6 the related $mCheck_{\text{NatATL}_\text{R}}$ algorithm. Regarding the complexity, since we use an oracle over a non-deterministic algorithm we have that checking the model checking problem is $\mathbf{NP^{NP}} = \mathbf{\Sigma_2^P}$. □

Theorem 5.5.3 *Model checking* NatATL_R *with variable* k *is in* $\mathbf{\Delta_3^P}$.

```
1  Algorithm mCheck_{NatATL_R}(M, q, φ, k):
2  Guess s_A with compl(s_A) ≤ k
3  t = Oracle(M, q, φ, s_A)
4  return (¬t)
```

Figure 5.6: Model checking NatATL_R with variable k

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

Figure 5.7: Summary of model checking complexity results

Proof. We can use a bottom-up procedure similarly to the one we have used in the proof of Theorem 5.3.1 for NatATL$_{\mathrm{r}}$, by looping the construction in Theorem 5.5.2. In this case, we use an oracle over a non-deterministic procedure over a polynomial procedure, so we obtain that the overall complexity to solve the addressed problem is $\mathbf{P^{NP^{NP}}} = \mathbf{\Delta_3^P}$. □

5.6 Summary and Future Work

In this chapter, we propose an alternative take on strategic reasoning, where agents can handle only relatively simple strategies. We use a natural representation of strategies by lists of guarded actions, and assume that only strategies up to size k can be employed as witnesses to formula $\langle\langle A\rangle\rangle^{\leq k}\gamma$. In terms of technical results, we show that model-checking for NatATL with memoryless strategies is in $\mathbf{P}$ when k is fixed, and $\mathbf{\Delta_2^P}$-complete when k is a parameter of the problem. For strategies with recall, the problem is in $\mathbf{\Delta_2^P}$ when k is fixed, and in $\mathbf{\Delta_3^P}$ in the general case, cf. the summary presented in Figure 5.7.

Clearly, reasoning about *simple* natural memoryless strategies is no more difficult than about arbitrary ATL strategies (and in practice we expect it to be actually easier). On the other hand, verification of natural strategies with recall seems distinctly harder. It would be interesting to look for conditions under which the latter kind of strategies can be synthesized in polynomial time.

We also prove an important property that sets NatATL apart from standard ATL: in NatATL, the memoryless and memoryfull semantics do not coincide.

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

temporal complexity via timing constraints in the vein of [BLMO07, AJK+17].

Bibliography

[AAK15] S. Almagor, G. Avni, and O. Kupferman. Repairing Multi-Player Games. In *CONCUR*, pages 325–339, 2015.

[ABL07] M. Arenas, P. Barceló, and L. Libkin. Combining Temporal Logics for Querying XML Documents. In *ICDT*, pages 359–373, 2007.

[ACY95] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing Tests for Nondeterministic and Probabilistic Machines. In *STOC*, pages 363–372, 1995.

[ADLM07] N. Alechina, M. Dastani, B. Logan, and J.-J. Ch. Meyer. A Logic of Agent Programs. In *AAAI*, pages 795–800, 2007.

[Ågo06] T. Ågotnes. Action and Knowledge in Alternating-time Temporal Logic. *Synthese*, 149(2):377–409, 2006.

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *JACM*, 49(5):672–713, 2002.

[AJK+17] E. Andre, W. Jamroga, M. Knapik, W. Penczek, and L. Petrucci. Timed ATL: Forget Memory, Just Count. In *AAMAS*, pages 1460–1462, 2017.

[AKH02] E. Altman, H. Kameda, and Y. Hosokawa. Nash Equilibria in Load Balancing in Distributed Computer Systems. *IGTR*, 4(2):91–100, 2002.

[ALDM08] N. Alechina, B. Logan, M. Dastani, and J.-J. Ch. Meyer. Reasoning about Agent Execution Strategies. In *AAMAS*, pages 1455–1458, 2008.

[ALM+13] B. Aminof, A. Legay, A. Murano, O. Serre, and M. Y. Vardi. Pushdown Module Checking with Imperfect Information. *IC*, 223:1–17, 2013.

[ALMS08] B. Aminof, A. Legay, A. Murano, and O. Serre. μ-calculus Pushdown Module Checking with Imperfect State Information. In *IFIP-TCS*, pages 333–348, 2008.

[ALNR09] N. Alechina, B. Logan, N.H. Nga, and A. Rakib. A Logic for Coalitions with Bounded Resources. In *IJCAI*, pages 659–664, 2009.

[ALNR10] N. Alechina, B. Logan, H.N. Nguyen, and A. Rakib. Resource-Bounded Alternating-Time Temporal Logic. In *AAMAS*, pages 481–488, 2010.

[AMMR16] B. Aminof, V. Malvone, A. Murano, and S. Rubin. Graded Strategy Logic: Reasoning about Uniqueness of Nash Equilibria. In *AAMAS*, pages 698–706, 2016.

[AMMR17] B. Aminof, V. Malvone, A. Murano, and S. Rubin. Graded Modalities in Strategy Logic. *IC*, 2017. To appear.

[AMR15] B. Aminof, A. Murano, and S. Rubin. On CTL* with Graded Path Modalities. In *LPAR*, pages 281–296, 2015.

[ATO+09] T. Antal, A. Traulsen, H. Ohtsuki, C.E. Tarnita, and M.A. Nowak. Mutation-Selection Equilibrium in Games with Multiple Strategies. *JTB*, 258(4):614–622, 2009.

[AV10] I. F. Akyildiz and M. C. Vuran. *Wireless sensor networks*. John Wiley & Sons, 2010.

[AW09] T. Agotnes and D. Walther. A Logic of Strategic Ability Under Bounded Memory. *JLLI*, 18(1):55–77, 2009.

[Bár15] E. Bárcenas. A Counting Logic for Trees. *Computación y Sistemas*, 19(2):407–422, 2015.

[BBF+10] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification: Model-Checking Techniques and Tools.* Springer, 2010.

[BBL15] F. Baader, S. Borgwardt, and M. Lippmann. Temporal conjunctive queries in expressive description logics with transitive roles. In *AJCAI*, pages 21–33, 2015.

[BBV86] T. C. Bergstrom, L. E. Blume, and H. R. Varian. On the Private Provision of Public Goods. *JPE*, 29(1):25–49, 1986.

[BC03] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM*, pages 1713–1723, 2003.

[BCJK15] R. Bloem, K. Chatterjee, S. Jacobs, and R. Könighofer. Assume-guarantee synthesis for concurrent reactive programs with partial information. In *TACAS*, pages 517–532, 2015.

[BCS08] M. Barlo, G. Carmona, and H. Sabourian. Bounded memory with finite action spaces. Technical report, Sabanci University, Universidade Nova de Lisboa and University of Cambridge, 2008.

[Bel15] F. Belardinelli. A Logic of Knowledge and Strategies with Imperfect Information. In *LAMAS*, 2015.

[BF10a] N. Bulling and B. Farwer. Expressing Properties of Resource-Bounded Systems: The Logics RTL* and RTL. In *CLIMA*, pages 22–45, 2010.

[BF10b] N. Bulling and B. Farwer. On the (Un-)Decidability of Model Checking Resource-Bounded Agents. In *ECAI*, pages 567–572, 2010.

[BFVW06] R. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying Multi-Agent Programs by Model Checking. *AAMAS*, 12(2):239–256, 2006.

[BGM15] P. Bouyer, P. Gardy, and N. Markey. Weighted Strategy Logic with Boolean Goals Over One-Counter Games. In *FSTTCS*, pages 69–83, 2015.

[Bin92] K. G. Binmore. *Fun and Games: A Text on Game Theory*. D.C. Heath, 1992.

[BJ14] N. Bulling and W. Jamroga. Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. *AAMAS*, 28(3):474–518, 2014.

[BK10] D. Berwanger and L. Kaiser. Information Tracking in Games on Graphs. *JLLI*, 19(4):395–412, 2010.

[BKR10] V. Bárány, L. Kaiser, and A. M. Rabinovich. Expressing Cardinality Quantifiers in Monadic Second-Order Logic over Trees. *FI*, 100(1-4):1–17, 2010.

[BLLM09] T. Brihaye, A. Da Costa Lopes, F. Laroussinie, and N. Markey. ATL with Strategy Contexts and Bounded Memory. In *LFCS*, pages 92–106, 2009.

[BLMO07] T. Brihaye, F. Laroussinie, N. Markey, and G. Oreiby. Timed Concurrent Game Structures. In *CONCUR*, pages 445–459, 2007.

[BLMR17] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of Multi-agent Systems with Imperfect Information and Public Actions. In *AAMAS*, pages 1268–1276, 2017.

[BLMV08] P.A. Bonatti, C. Lutz, A. Murano, and M.Y. Vardi. The Complexity of Enriched Mu-Calculi. *LMCS*, 4(3):1–27, 2008.

[BMM09] A. Bianco, F. Mogavero, and A. Murano. Graded Computation Tree Logic. In *LICS*, pages 342–351, 2009.

[BMM10] A. Bianco, F. Mogavero, and A. Murano. Graded Computation Tree Logic with Binary Coding. In *CSL*, pages 125–139, 2010.

[BMM12] A. Bianco, F. Mogavero, and A. Murano. Graded Computation Tree Logic. *TOCL*, 13(3):25:1–25:53, 2012.

[BMM17] R. Berthon, B. Maubert, and A. Murano. Decidability Results for ATL* with Imperfect Information and Perfect Recall. In *AAMAS*, pages 1250–1258, 2017.

[BMMRV17] R. Berthon, B. Maubert, A. Murano, S.Rubin, and M. Y. Vardi. Strategy logic with imperfect information. In *LICS*, pages 1–12, 2017.

[BMP10] L. Bozzelli, A. Murano, and A. Peron. Pushdown module checking. *FMSD*, 36(1):65–95, 2010.

[CD14] K. Chatterjee and L. Doyen. Partial-observation stochastic games: How to win when belief fails. *TOCL*, 15(2):16, 2014.

[CDHR07] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.F. Raskin. Algorithms for omega-regular games with imperfect information. *LMCS*, 3(4):1–23, 2007.

[CDL99] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *IJCAI*, pages 84–89, 1999.

[CE81] E.M. Clarke and E.A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *LP*, pages 52–71, 1981.

[CEO14] D. Calvanese, T. Eiter, and M. Ortiz. Answering Regular Path Queries in Expressive Description Logics via Alternating Tree-Automata. *IC*, 237:12–55, 2014.

[CGLV10] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Node Selection Query Languages for Trees. In *AAAI*, 2010.

[CGP02] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 2002.

[CH12] K. Chatterjee and T. A. Henzinger. A survey of stochastic omega-regular games. *JCSS*, 78(2):394–413, 2012.

[CHP07] K. Chatterjee, T.A. Henzinger, and N. Piterman. Strategy Logic. In *CONCUR*, pages 59–73, 2007.

[CHP10] K. Chatterjee, T.A. Henzinger, and N. Piterman. Strategy Logic. *IC*, 208(6):677–693, 2010.

[CHS99] R. Cornes, R. Hartley, and T. Sandler. An Elementary Proof via Contraction. *JPET*, 1(4):499–509, 1999.

[ČLM15] P. Čermák, A. Lomuscio, and A. Murano. Verifying and Synthesising Multi-Agent Systems against One-Goal Strategy Logic Specifications. In *AAAI*, pages 2038–2044, 2015.

[ČLMM14] P. Čermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications. In *CAV*, pages 524–531, 2014.

[CS08] V. Conitzer and T. Sandholm. New complexity results about nash equilibria. *Games and Economic Behavior*, 63(2):621–641, 2008.

[CTC07] M. Chang, Y. Tseng, and J. Chen. A Scenario Planning Approach for the Flood Emergency Logistics Preparation Problem under Uncertainty. *LTR*, 43(6):737–754, 2007.

[DB16] H. Duijf and J. M. Broersen. Representing Strategies. In *SR*, pages 15–26, 2016.

[DJ10] M. Dastani and W. Jamroga. Reasoning about Strategies of Multi-Agent Programs. In *AAMAS*, pages 625–632, 2010.

[DT11] C. Dima and F. L. Tiplea. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. *CoRR*, abs/1102.4225, 2011.

[EH86] E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time. *JACM*, 33(1):151–178, 1986.

[EJ88] E.A. Emerson and C.S. Jutla. The Complexity of Tree Automata and Logics of Programs (Extended Abstract). In *FOCS*, pages 328–337, 1988.

[EJ91] E.A. Emerson and C.S. Jutla. Tree Automata, μ-Calculus and Determinacy. In *FOCS*, pages 368–377, 1991.

[EJ99] E.A. Emerson and C.S. Jutla. The Complexity of Tree Automata and Logics of Programs. *SJM*, 29(1):132–158, 1999.

[FAGV12] F.Mogavero, A.Murano, G.Perelli, and M.Y. Vardi. What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic. In *CONCUR*, pages 193–208, 2012.

[FE15] C. Feier and T. Eiter. Reasoning with Forest Logic Programs Using Fully Enriched Automata. In *LPNMR*, pages 346–353, 2015.

[Fin72] K. Fine. In So Many Possible Worlds. *NDJFL*, 13(4):516–520, 1972.

[FMP08] A. Ferrante, A. Murano, and M. Parente. Enriched Mu-Calculi Module Checking. *LMCS*, 4(3):1–21, 2008.

[FNP09a] A. Ferrante, M. Napoli, and M. Parente. Graded-CTL: Satisfiability and Symbolic Model Checking. In *ICFEM*, pages 306–325, 2009.

[FNP09b] A. Ferrante, M. Napoli, and M. Parente. Model Checking for Graded CTL. *FI*, 96(3):323–339, 2009.

[Fra92] C. D. Fraser. The Uniqueness of Nash Equilibrium in the Private Provision of Public Goods: an Alternative Proof. *JPE*, 49(3):389–390, 1992.

[FS05] E. Faingold and Y. Sannikov. Equilibrium degeneracy and reputation effects in continuous time games. Technical report, mimeo, 2005.

[GHW15] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Expresiveness and Complexity Results for Strategic Reasoning. In *CONCUR*, pages 268–282, 2015.

[GHW17] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Reasoning about equilibria in game-like concurrent systems. *APAL*, 168(2):373–403, 2017.

[GK93] A. Glazer and K. A. Konrad. Private Provision of Public Goods, Limited Tax Deducibility, and Crowding Out. *PFA*, 50(2):203–216, 1993.

[GLLS07] O. Grumberg, M. Lange, M. Leucker, and S. Shoham. When not losing is better than winning: Abstraction and refinement for the full mu-calculus. *IC*, 205(8):1130–1148, 2007.

[GMP+17] J. Gutierrez, A. Murano, G. Perelli, S. Rubin, and M. Wooldridge. Nash Equilibria in Concurrent Games with Lexicographic Preferences. In *IJCAI*, pages 1067–1073, 2017.

[GOR97] E. Grädel, M. Otto, and E. Rosen. Two-Variable Logic with Counting is Decidable. In *LICS*, pages 306–317, 1997.

[GSW14] A. Gupta, S. Schewe, and D. Wojtczak. Making the best of limited memory in multi-player discounted sum games. *arXiv preprint arXiv:1410.4154*, 2014.

[GZ89] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *GEB*, 1(1):80–93, 1989.

[HB91] B. Hollunder and F. Baader. Qualifying Number Restrictions in Concept Languages. In *KR*, pages 335–346, 1991.

[HK82] D. Harel and D. Kozen. Process Logic: Expressiveness, Decidability, Completeness. *JCSS*, 25(2):144–170, 1982.

[HLMW14] A. Herzig, E. Lorini, F. Maffre, and D. Walther. Alternating-time Temporal Logic with Explicit Programs. In *LAMAS*, 2014.

[HO09] J. Hörner and W. Olszewski. How robust is the Folk Theorem? *QJE*, 124(4):1773–1814, 2009.

[HP85] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and models of concurrent systems. NATO Advanced Summer Institutes vol. F-13.*, pages 477–498. Springer, 1985.

[HS04] I. Horrocks and U. Sattler. Decidability of SHIQ with Complex Role Inclusion Axioms. *AI*, 160(1-2):79–104, 2004.

[Imm81] N. Immerman. Number of Quantifiers is Better Than Number of Tape Cells. *JCSS*, 22(3):384–406, 1981.

[JÅ07] W. Jamroga and T. Ågotnes. Constructive knowledge: what agents can achieve under imperfect information. *JANCL*, 17(4):423–475, 2007.

[JD06] W. Jamroga and J. Dix. Model Checking ATL$_{ir}$ is Indeed Δ_2^P-complete. In *EUMAS*, 2006.

[JDW02] J.Bernet, D.Janin, and I. Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO*, 36(3):261–275, 2002.

[JM14] W. Jamroga and A. Murano. On Module Checking and Strategies. In *AAMAS*, pages 701–708, 2014.

[JM15] W. Jamroga and A. Murano. Module Checking of Strategic Ability. In *AAMAS*, pages 227–235, 2015.

[JMM17] W. Jamroga, V. Malvone, and A. Murano. Reasoning about Natural Strategic Ability. In *AAMAS*, pages 714–722, 2017.

[JvdH04] W. Jamroga and W. van der Hoek. Agents that Know How to Play. *FI*, 63(2-3):185–219, 2004.

[Kel76] R.M. Keller. Formal verification of parallel programs. *CACM*, 19(7):371–384, 1976.

[Kit00] H. Kitano. Robocup Rescue: A Grand Challenge for Multi-Agent Systems. In *IEEE*, pages 5–12, 2000.

[KM98] M. Kandori and H. Matsushima. Private observation, communication and collusion. *Econometrica*, 66(3):627–652, 1998.

[KPV14] O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with Rational Environments. In *EUMAS*, pages 219–235, 2014.

[Kri63] S.A. Kripke. Semantical Considerations on Modal Logic. *APF*, 16:83–94, 1963.

[KSV02] O. Kupferman, U. Sattler, and M.Y. Vardi. The Complexity of the Graded μ-Calculus. In *CADE*, pages 423–437, 2002.

[KT01] H. Kitano and S. Tadokoro. Robocup Rescue: A Grand Challenge for Multiagent and Intelligent Systems. *AI magazine*, 22(1):39, 2001.

[KTN+99] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. Robocup rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research. In *IEEE*, pages 739–743, 1999.

[KV96] O. Kupferman and M.Y. Vardi. Module Checking. In *CAV*, pages 75–86, 1996.

[KV97] O. Kupferman and M. Y. Vardi. Module Checking Revisited. In *CAV*, pages 36–47, 1997.

[KV00] O. Kupferman and M. Y. Vardi. Synthesis with incomplete informatio. In *Advances in Temporal Logic*, pages 109–127. Springer, 2000.

[KVW00] O. Kupferman, M.Y. Vardi, and P. Wolper. An Automata Theoretic Approach to Branching-Time Model Checking. *JACM*, 47(2):312–360, 2000.

[KVW01] O. Kupferman, M. Y. Vardi, and P. Wolper. Module Checking. *IC*, 164(2):322–344, 2001.

[LBS08] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction (Synthesis Lectures on Artificial Intelligence and Machine Learning)*. M&C, 2008.

[LLM10] A.D.C. Lopes, F. Laroussinie, and N. Markey. ATL with Strategy Contexts: Expressiveness and Model Checking. In *FSTTCS*, pages 120–132, 2010.

[LMO08] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *arXiv preprint arXiv:0804.2435*, 2008.

[LQR09] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *CAV*, pages 682–688, 2009.

[LR06a] A. Lomuscio and F. Raimondi. MCMAS: A Model Checker for Multi-agent Systems. In *TACAS*, pages 450–454, 2006.

[LR06b] A. Lomuscio and F. Raimondi. Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems. In *AAMAS*, pages 161–168, 2006.

[LST05] C. Lutz, U. Sattler, and L. Tendera. The Complexity of Finite Model Reasoning in Description Logics. *IC*, 199(1):132–171, 2005.

[LWW07] C. Lutz, D. Walther, and F. Wolter. Conservative Extensions in Expressive Description Logics. In *IJCAI*, pages 453–458, 2007.

[Mar75] A.D. Martin. Borel Determinacy. *AM*, 102(2):363–371, 1975.

[Mar85] A.D. Martin. A Purely Inductive Proof of Borel Determinacy. In *SPM*, pages 303–308, 1985.

[MMMS15] V. Malvone, F. Mogavero, A. Murano, and L. Sorrentino. On the Counting of Strategies. In *TIME*, pages 170–179, 2015.

[MMMS17] V. Malvone, F. Mogavero A. Murano, and L. Sorrentino. Reasoning about Graded Strategy Quantifiers. *IC*, 2017. To appear.

[MMPV14] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning About Strategies: On the Model-Checking Problem. *TOCL*, 15(4):34, 2014.

[MMS14a] F. Mogavero, A. Murano, and L. Sauro. A Behavioral Hierarchy of Strategy Logic. In *CLIMA*, pages 148–165, 2014.

[MMS14b] F. Mogavero, A. Murano, and L. Sauro. Strategy Games: A Renewed Framework. In *AAMAS*, pages 869–876, 2014.

[MMS15] V. Malvone, A. Murano, and L. Sorrentino. Games with additional winning strategies. In *CILC*, pages 175–180, 2015.

[MMS16] V. Malvone, A. Murano, and L. Sorrentino. Hiding Actions in Concurrent Games. In *ECAI*, pages 1686–1687, 2016.

[MMS17a] V. Malvone, A. Murano, and L. Sorrentino. Additional Winning Strategies in Reachability Games. *FI*, 2017. To appear.

[MMS17b] V. Malvone, A. Murano, and L. Sorrentino. Hiding Actions in Multi-Player Games. In *AAMAS*, pages 1205–1213, 2017.

[MMV10a] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning About Strategies. In *FSTTCS*, pages 133–144, 2010.

[MMV10b] F. Mogavero, A. Murano, and M.Y. Vardi. Relentful Strategic Reasoning in Alternating-Time Temporal Logic. In *LPAR*, pages 371–387, 2010.

[MNP08] A. Murano, M. Napoli, and M. Parente. Program Complexity in Hierarchical Module Checking. In *LPAR*, pages 318–332, 2008.

[Mog11] F. Mogavero. *Logics in Computer Science.* PhD thesis, Università degli Studi di Napoli "Federico II", Napoli, Italy, 2011.

[MP15] A. Murano and G. Perelli. Pushdown Multi-Agent System Verification. In *IJCAI*, pages 1090–1097, 2015.

[MS95] D.E. Muller and P.E. Schupp. Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of Theorems of Rabin, McNaughton, and Safra. *TCS*, 141(1-2):69–107, 1995.

[MS15] A. Murano and L. Sorrentino. A Game-Based Model for Human-Robots Interaction. In *WOA*, pages 146–150, 2015.

[Mye91] R.B. Myerson. *Game Theory: Analysis of Conflict.* Harvard University Press, 1991.

[NJ09] P. Novák and W. Jamroga. Code Patterns for Agent Oriented Programming. In *AAMAS*, pages 105–112, 2009.

[OR94] M.J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[ORS93] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *NET*, 1(5):510–521, 1993.

[Pav12] L. Pavel. *Game Theory for Control of Optical Networks*. Springer, 2012.

[PC79] G.P. Papavassilopoulos and J. B. Cruz. On the Uniqueness of Nash Strategies for a Class of Analytic Differential Games. *JOTA*, 27(2):309–314, 1979.

[Pnu77] A. Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57, 1977.

[PP04] D. Perrin and J. Pin. *Infinite Words.* Elsevier, 2004.

[PR89] A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *ACM*, pages 179–190, 1989.

[PR90] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *FOCS*, pages 746–757, 1990.

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *SP*, pages 337–351, 1982.

[Rab69] M.O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *TAMS*, 141:1–35, 1969.

[RAM15] S. Rubin, B. Aminof, and A. Murano. On CTL* with Graded Path Modalities. In *LPAR*, pages 281–296, 2015.

[Rei84] J. H. Reif. The Complexity of Two-Player Games of Incomplete Information. *JCSS*, 29(2):274–301, 1984.

[RES+10] S. Roy, C. Ellis, S. G. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu. A Survey of Game Theory as Applied to Network Security. In *HICSS*, pages 1–10, 2010.

[San07] Y. Sannikov. Games with imperfectly observable actions in continuous time. *Econometrica*, 75(5):1285–1329, 2007.

[Sch02] P. Schnoebelen. The Complexity of Temporal Logic Model Checking. In *AIML*, pages 393–436, 2002.

[Sch04] P.Y. Schobbens. Alternating-Time Logic with Imperfect Recall. *ENTCS*, 85(2):82–93, 2004.

[Sel65] R. Selten. Spieltheoretische Behandlung eines Oligopolmodells mit Nachfragetragheit. *ZgS*, 121(2):301–324, 1965.

[Sip06] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology Boston, 2006.

[SLCB13] D. Simchi-Levi, X. Chen, and J. Bramel. *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management*. Springer, 2013.

[Smi82] J. M. Smith. *Evolution and the Theory of Games*. Cambridge university press, 1982.

[Tho90] W. Thomas. Automata on Infinite Objects. In *Handbook of Theoretical Computer Science (vol. B)*, pages 133–191. MIT Press, 1990.

[Umm06] M. Ummels. Rational Behaviour and Strategy Construction in Infinite Multiplayer Games. In *FSTTCS*, pages 212–223, 2006.

[vdHJW05] W. van der Hoek, W. Jamroga, and M. Wooldridge. A Logic for Strategic Reasoning. In *AAMAS*, pages 157–164, 2005.

[vdHM92] W. van der Hoek and J.J. Meyer. Graded modalities in epistemic logic. In *LFCS*, pages 503–514, 1992.

[vdHW02] W. van der Hoek and M.J. Wooldridge. Tractable Multiagent Planning for Epistemic Goals. In *AAMAS*, pages 1167–1174, 2002.

[vdMW05] R. van der Meyden and T. Wilke. Synthesis of Distributed Systems from Knowledge-Based Specifications. In *CONCUR*, pages 562–576, 2005.

[Ves13] S. Vester. Alternating-time temporal logic with finite-memory strategies. In *GandALF*, pages 194–207, 2013.

[VW86] M.Y. Vardi and P. Wolper. Automata-Theoretic Techniques for Modal Logics of Programs. *JCSS*, 32(2):183–221, 1986.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages (An Introduction).* MIT Press, 1993.

[WJ95] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *KER*, 10(2):115–152, 1995.

[Woo02] M. Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons, 2002.

[WvdHW07] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-Time Temporal Logic with Explicit Strategies. In *TARK*, pages 269–278, 2007.

[YS12] N. Yadav and S. Sardiña. Reasoning about Agent Programs Using ATL-Like Logics. In *JELIA*, pages 437–449, 2012.

[ZG11] Y. Zhang and M. Guizani. *Game Theory for Wireless Communications and Networking*. CRC Press, 2011.

List of Figures